TENSOR DECOMPOSITIONS AND RANK APPROXIMATION OF TENSORS WITH APPLICATIONS

by

RAMIN GOUDARZI KARIM

CARMELIZA NAVASCA, ADVISOR
COMMETTEE MEMBERS:
BRENDAN AMES
IAN KNOWLES
DA YAN
GUO-HUI ZHANG

A DISSERTATION

Submitted to the graduate faculty of The University of Alabama, The University of Alabama at Birmingham, and The University of Alabama in Huntsville in partial fulfillment of the requirements for the degree of Doctor of Philosophy

BIRMINGHAM, ALABAMA

2019

ABSTRACT

TENSOR DECOMPOSITIONS AND RANK APPROXIMATION OF TENSORS VIA SPARSE OPTIMIZATION

RAMIN GOUDARZI KARIM

Tensor decompositions are higher-order analogues of matrix decompositions which have applications in data analysis, signal processing, machine learning and data mining. One of the most challenging problems in the tensor decomposition area is to approximate the rank of a given tensor. Unlike the matrix case there is no simple formula to bound the rank of a tensor. In fact, finding the exact rank of a tensor is an NP hard problem. In this thesis we formulate the tensor rank estimation of a tensor as an optimization problem and estimate the rank via ℓ_1 minimization. We propose a numerical iterative method based on the proximal alternating minimization algorithm and discuss the required conditions for the global convergence of the algorithm. The performance of our algorithm is tested on several types of data such as randomly generated tensors, RGB images and surveillance videos in order to separate the background and foreground of them.

In addition, this thesis studies the block sampling of the alternating least squares technique (ALS) and proposes an effective method for the CP decomposition of tensors. The method is tested on randomly generated data as well as real data. The proposed method converges faster that ALS in some cases when there is a presence of swamp. In addition, it requires less computations in each iteration. Also, the application of CP decomposition in image compression is provided.

DEDICATION

TO MY BELOVED PARENTS

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Carmeliza Navasca for the continuous support of my PhD study and related research, for hrs patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Guo-Hui Zhang, Dr. Da Yan, Dr. Ian Knowles Dr. Brendan Ames, for their insightful comments and encouragement, but also for the hard questions which incented me to widen my research from various perspectives.

Also, I would like to express deep gratitude to UAB Mathematics Department to provide a helpful environment. I extend my deep gratitude to UAB Mathematics Department members Dr. Weikard, Dr. Stolz, Dr. Karpeshina, Dr. Zeng, Dr. Starr, Dr. Nkashama and UAB Computer Science Department members Dr. Zheng, Dr. Sprague for their knowledge, the support, and the encouragement.

Last but not the least, I would like to thank my family: my parents and to my sister for supporting me spiritually throughout writing this thesis and my my life in general.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure *Page*

CHAPTER 1

Introduction

The CANDECOMP/PARAFAC (CP) decomposition of multi-dimensional arrays was first introduced by Hitchcock in 1927 [30, 31]. It breaks down a tensor into a sum of simpler tensors which can be seen as a similar method to the matrix singular value decomposition (SVD). However, unlike the matrix SVD, the CP decomposition of a tensor is more complex. The CP decomposition has received much attention in the different areas of science and engineering such as signal processing, neuroscience, data science and machine learning [61, 62, 2, 22]. In the last decade, CP decomposition and Tucker decomposition of tensors have been studied in the framework of numerical linear algebra, multilinear algebra and numerical analysis [19, 17].

Given a positive integer R, the number of rank one components in the CP decomposition, there are several of algorithms to find a CP decomposition. Our focus for computing the CP decomposition is on alternating least squares (ALS). It was first proposed by Carroll and Chang [15] and has been used widely due to its simplicity and ease of implementation. However, it can take so many iterations to converge and it is not guaranteed to converge to a global minimum. Also, initial guess plays a key role in the convergence of ALS [72]. There are several papers which discuss the this problem and propose algorithms to improve the efficiency of the alternating least squares method [49, 51].

One of the most important issues dealing with CP decomposition of a tensor is to determine the rank of a tensor. The rank determination of a tensor is an NP hard problem [28]. There are several papers discussing the difficulties estimating the rank of tensors and suggesting new definitions like maximum attainable rank, border rank and typical rank of tensors [68, 69]. Moreover, most of the suggested algorithms do

not provide an estimation on the tensor rank. Most of the numerical methods for computing CP decomposition require the rank of a given tensor. In addition, they begin with a lower guess for the tensor rank R and increase it to find a perfect fit for the residual. Ideally, for noise free data the procedure is to compute the CP for $R = r, r+1, \ldots$ rank one components and stop when a good residual error is acquired. In the presence of noise, the problem is even more complex because having a good fit does not guarantee the rank in this case. However, this method can cause several computing problems. One of the problems that may occur is the case when the tensor is degenerate. The degeneracy of tensor is due to the ill-posedness of the CP formulation.

This thesis is focused on finding a approximation of the tensor rank and its CP decomposition. The tensor rank problem can be formulated as an optimization problem

$$\min_{\alpha} \|\alpha\|_0 \quad \text{s.t.} \quad \mathcal{X} = [\alpha, A, B, C]_R,$$

where $\|, \|_0$ represents the number of nonzero elements of a vector. However, the ℓ_0 norm is not convex. Therefore it is not an actual norm.Inspired by compressive sensing we work with the ℓ_1 regularization. It is known from compressive sensing that the minimization of ℓ_1 norm of α yields the sparse solution of the corresponding linear system. The ℓ_1 regularization term also imposes the boundedness of α and prevents the norm of the solution to approach to infinity. This resolves the ill-posedness of the CP problem. Since the objective function is a sum of a smooth and non-smooth functions, we adopt methods involving the proximal linearization of the objective function.

In chapter 2, we review the basic definitions and terminologies in the area of tensor decompositions. Most of the notations and definitions are adopted from [36]. There are different methods for vectorization and matricization of a tensor which appeared in the publications. However, the order of the array does not affect the results as long as we remain consistent through the calculation.

In chapter 3, we discuss the necessary concepts of optimization and review the basic algorithms which will be used in the following chapters. Most of the optimization techniques which are used in tensor decompositions are the block coordinate descent methods of Gauss-Seidel type. The use of proximal algorithms in tensor decompositions has expanded in recent years [79].

In chapter 4, we formulate the CP decomposition as an optimization problem in order to find the CP decomposition. The ALS algorithm will be reviewed in the chapter as well as non linear methods. The performance of different methods are discussed and compared for variety of tensors.

In chapter 5, we propose an algorithm to approximate the rank of a given tensor. Most of the results are established for third order tensor. However, it can be extended to any order. The technique is based on ℓ_1 minimization. We also test the method on surveillance videos and images. Our experiments show a good improvement to the previous works.

In chapter 6, the block sampling ALS method is proposed. The sampling shows an improvement in the convergence of ALS when swamp happens. I order to test the method, we apply the algorithm to an RGB image.

CHAPTER 2

Introduction to Tensors and Tensor Decompositions

2.1. Tensor Definition and Preliminaries

An Nth order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is a real N-dimensional array where its element is denoted by

$$[\mathcal{X}]_{i_1 \ldots i_N} = x_{i_1 \ldots i_N}, \qquad 1 \leq i_j \leq I_j, \quad 1 \leq j \leq N$$

The order of a tensor is the number of dimensions. By the definition above, vectors are tensors of order one and matrices are tensors of order two. If $I_1 = I_2 = \ldots = I_N$, then the tensor $\mathcal{X}$ is called cubic (cubical).

EXAMPLE 2.1.1. *An order-zero tensor is a scalar. A third order tensor has three indices, i, j and k as illustrated in the Figure 2.1. The element x_{ijk} is located on the i-th row, j-th column and k-th tube of the cube in the figure.*

DEFINITION 2.1.1. *An Nth order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is called diagonal if $x_{i_1 i_2 \ldots i_N} \neq 0$ only if $i_1 = i_2 = \ldots = i_N$. Figure 2.2 illustrates a cubic tensor with a constant value c along its diagonal.*

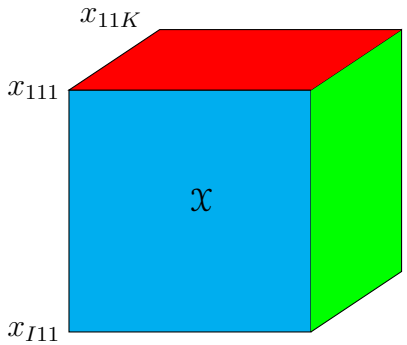

FIGURE 2.1. A third order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$.

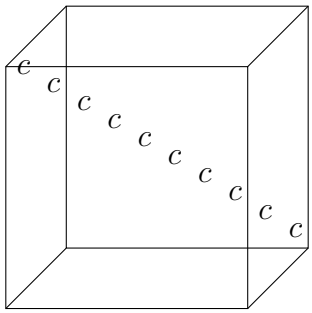

FIGURE 2.2. A cubical order three diagonal tensor with a constant value c along its diagonal.

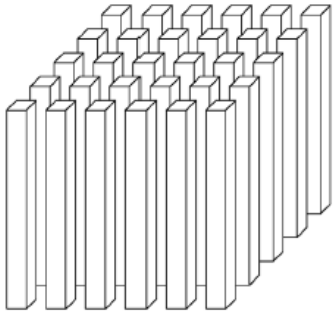

FIGURE 2.3. Mode-1(column) fibers $x_{:jk}$.

Fibers of a tensor are the higher-order analogue of matrix rows and columns. A fiber is obtained from a tensor by fixing every index but one. For instance, the columns of a matrix are mode-1 fibers and the rows are mode-2 fibers. Third-order tensors have column, row and tube fibers which we denote them as $x_{:jk}, x_{i:k}$ and $x_{ij:}$, respectively. Figures (2.3), (2.4) and (2.5) show the mode-1 (column), mode-2 (row) and mode-3 (tube) fibers of $\mathcal{X}$.

Slices are two-dimensional sections of a tensor, which are obtained by fixing all but two indices. The horizontal, lateral and frontal slices of a third order tensor $\mathcal{X}$ are denoted by $X(i,:,:), X(:,j,:)$ and $X(:,:,k)$ respectively. Figures (2.6), (2.7) and (2.8) show the horizontal, lateral and frontal slices of a third order tensor $\mathcal{X}$.

DEFINITION 2.1.2. *The mode-n matricization (unfolding) of a tensor* $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ *is shown by* $X_{(n)}$ *and is defined as follow: The* $(i_1, \ldots i_N)$ *element of tensor maps to*

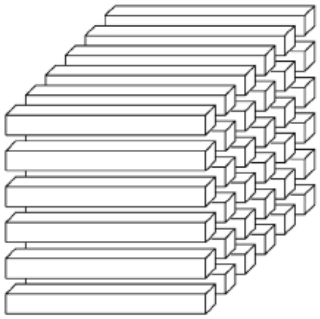

FIGURE 2.4. Mode-2(row) fibers $x_{i:k}$.

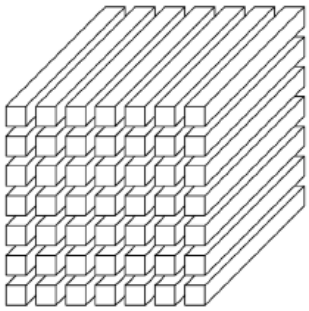

FIGURE 2.5. Mode-3(tube) fibers $x_{ij:}$

matrix element (i_n, j) *where*

$$j = 1 + \sum_{k \neq n}^{N} (i_k - 1) J_k, \quad J_k = \prod_{m \neq n}^{k-1} I_m$$

EXAMPLE 2.1.2. *Let* $\mathcal{X} \in \mathbb{R}^{4 \times 3 \times 2}$ *then the three mode-n unfoldings are*

$$X_{(1)} = \begin{pmatrix} x_{111} & x_{121} & x_{131} & x_{112} & x_{122} & x_{132} \\ x_{211} & x_{221} & x_{231} & x_{212} & x_{222} & x_{232} \\ x_{311} & x_{321} & x_{331} & x_{312} & x_{322} & x_{332} \\ x_{411} & x_{421} & x_{431} & x_{412} & x_{422} & x_{432} \end{pmatrix}$$

$$X_{(2)} = \begin{pmatrix} x_{111} & x_{211} & x_{311} & x_{411} & x_{112} & x_{212} & x_{312} & x_{412} \\ x_{121} & x_{221} & x_{321} & x_{421} & x_{122} & x_{222} & x_{322} & x_{422} \\ x_{131} & x_{231} & x_{331} & x_{431} & x_{132} & x_{232} & x_{332} & x_{432} \end{pmatrix}$$

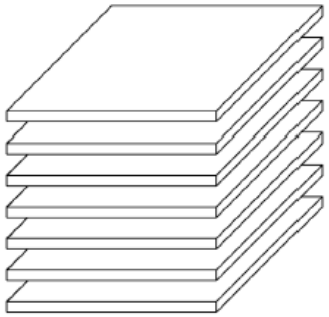

FIGURE 2.6. Horizontal slices of an order-three tensor $X(i,:,:)$.

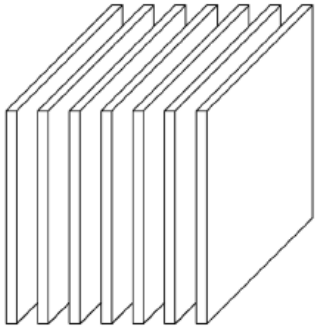

FIGURE 2.7. Lateral slices of an order-three tensor $X(:,j,:)$.

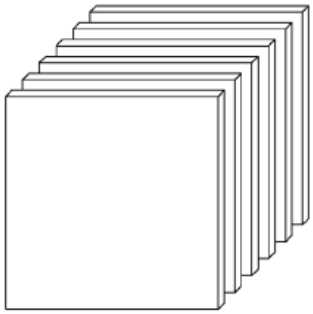

FIGURE 2.8. Frontal slices of an order-three tensor $X(:,:,k)$.

and

$$X_{(3)} = \begin{pmatrix} x_{111} & x_{211} & x_{311} & x_{411} & \dots & x_{131} & x_{231} & x_{331} & x_{431} \\ x_{112} & x_{212} & x_{312} & x_{412} & \dots & x_{132} & x_{232} & x_{332} & x_{432} \end{pmatrix}$$

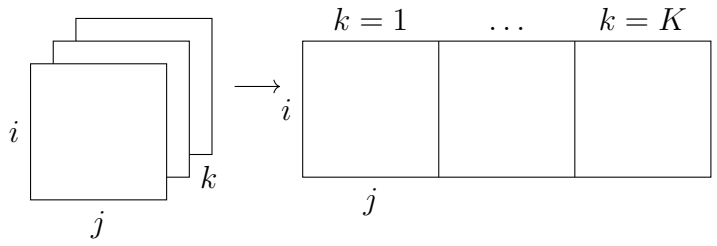

FIGURE 2.9. Mode-1 matricization of a third order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$.

In general, we can define an unfolding for $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ by choosing a set of row modes and a set of column modes, for example see [35]. Different orderings have been used in the literature for ordering the columns for the mode-n unfoldings, however the specific permutation of columns is not important as long as it remains consistent throughout the calculations [37]. Figure (2.9) shows the mode-1 unfolding of a third order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$.

DEFINITION 2.1.3. *The inner product of two same-sized tensors* $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ *is the sum of the products of their elements, i.e.,*

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1 \ldots i_N} x_{i_1 \ldots i_N} y_{i_1 \ldots i_N}.$$

This will define a norm of tensor (Frobenius) as follow:

$$\|\mathcal{X}\|_F = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle} = \left(\sum_{i_1 \ldots i_N} x_{i_1 \ldots i_N}^2 \right)^{1/2}.$$

This is analogous to the matrix Frobenius norm.

DEFINITION 2.1.4. *The outer product of* N *vectors* $a^{(1)} \in \mathbb{R}^{I_1}, \ldots, a^{(N)} \in \mathbb{R}^{I_N}$ *is an order-N tensor* $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ *whose elements are defined as*

$$x_{i_1 \ldots i_N} = a_{i_1}^{(1)} \ldots a_{i_N}^{(N)}, \qquad 1 \leq i_j \leq I_j$$

An order-N tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ is rank one if it can be written as the outer product of N vectors, i.e.,

$$\mathcal{X} = a^{(1)} \circ \ldots \circ a^{(N)}.$$

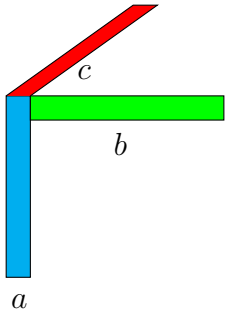

FIGURE 2.10. Schematic of a rank one tensor $a \circ b \circ c$.

Figure 2.10 shows the rank one third-order tensor $\mathcal{X}$, The (i, j, k) element of $\mathcal{X}$ is given by $a_i b_j c_k$.

$$\mathcal{X} = a \circ b \circ c.$$

DEFINITION 2.1.5. *An order N cubic tensor $\mathcal{X} \in \mathbb{R}^{I \times I \times \ldots \times I}$ is called symmetric if*

$$x_{i_{\sigma(1)} \ldots i_{\sigma(N)}} = x_{i_1 \ldots i_N}, \quad i_1, \ldots, i_N \in \{1, \ldots, n\},$$

for all permutations $\sigma : N \to N$.

For example, a third order tensor $\mathcal{X} \in \mathbb{R}^{I \times I \times I}$ is symmetric if

$$x_{ijk} = x_{ikj} = x_{jik} = x_{jki} = x_{kij} = x_{kji},$$

for all $i, j, k \in \{1, \ldots, I\}$.

DEFINITION 2.1.6. *(vectorization) Let $X \in \mathbb{R}^{m \times n}$ be an $m \times n$ matrix with a_i as the i-th column of X. The vec(.) operator turns X into a column vector as follows*

$$vec(X) = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}.$$

Likewise, we can define the the vectorization of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ by vectorizing the mode-1 matricization of $\mathcal{X}$, i.e.

$$\text{vec}(\mathcal{X}) = \text{vec}(X_{(1)}).$$

2.2. Some Tensor Operations and Matrix Multiplications

DEFINITION 2.2.1. *The Kronecker product of matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{K \times L}$ is denoted by $A \otimes B$ is a matrix of size $(IK) \times (JL)$ and is defined as*

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1J}B \\ a_{21}B & a_{22}B & \dots & a_{2J}B \\ \vdots & \vdots & & \vdots \\ a_{I1}B & a_{I2}B & \dots & a_{IJ}B \end{pmatrix}$$

PROPOSITION 2.1. *[75] The basic properties of Kronecker product*

(1) $(A \otimes B)^T = A^T \otimes B^T$

(2) $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$

(3) $(A \otimes B)(C \otimes D) = AC \otimes BD$

(4) $A \otimes (B \otimes C) = (A \otimes B) \otimes C$

DEFINITION 2.2.2. *[64] The Khatri-Rao product of two matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{K \times J}$ which is denoted by $A \odot B$, is a matrix of size $(IK) \times J$ and is the Kronecker product of the corresponding columns of A and B, i.e.,*

$$A \odot B = \begin{pmatrix} a_1 \otimes b_1 & \dots & a_J \otimes b_J \end{pmatrix}$$

where a_j and b_j represent the columns of A and B for $j = 1, \dots, J$.

When a and b are vectors then the definition of Khatri-Rao coincides with the definition of Kronecker product, i.e. $a \otimes b = a \odot b$.

DEFINITION 2.2.3. *The Hadamard product of two given matrices $A, B \in \mathbb{R}^{I \times J}$ is the product of corresponding elements. It is denoted by $A * B$ and is a matrix of size*

$I \times J$:

$$A * B = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2J}b_{2J} \\ \vdots & \vdots & & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \dots & a_{IJ}b_{IJ} \end{pmatrix}.$$

Hence the Hadamard product is an element-by-element product.

PROPOSITION 2.2. *Some properties of Hadamard product [64]*

(1) $A * B = B * A$

(2) $(A * B)^T = A^T * B^T$

(3) $A * (B * C) = (A * B) * C$

(4) $A * (B + C) = A * B + A * C$

(5) $(A \odot B)^T (A \odot B) = A^T A * B^T B$

DEFINITION 2.2.4. *(Moore-Penrose Inverse) Let* $A \in \mathbb{R}^{m \times n}$*, then the a pseudoinverse of* A *is defined as a matrix* $A^\dagger \in \mathbb{R}^{n \times m}$ *satisfying all of the following four conditions:*

(1) $AA^\dagger A = A$

(2) $A^\dagger A A^\dagger = A^\dagger$

(3) $(AA^\dagger)^T = AA^\dagger$

(4) $(A^\dagger A)^T = A^\dagger A$

It can be shown that the pseudoinverse of a matrix always exists and is unique [57]. When A has full rank, $A^\dagger$ can be expressed as a simple algebraic formula. If A is full column rank (the columns of A are linearly independent), then $A^T A$ is invertible therefore $A^\dagger = (A^T A)^{-1} A^T$ and when A is full row rank (the rows of A are linearly independent), then AA^T is invertible and $A^\dagger = A^T (AA^T)^{-1}$.

PROPOSITION 2.3. *Some properties of pseudoinverse*

(1) $(A^\dagger)^\dagger = A$

(2) $(A^T)^\dagger = (A^\dagger)^T$

(3) $(cA)^\dagger = c^{-1}A^\dagger, \quad c \neq 0$

(4) $(A \odot B)^\dagger = ((A^TA) * (B^TB))^\dagger (A \odot B)^T$

Let $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ be a rank one tensor obtained from outer product of N vectors, in particular assume

$$\mathcal{X} = a^{(1)} \circ \ldots \circ a^{(N)},$$

then there is a relation between the vectorization (matricization) of $\mathcal{X}$ and the Kronecker product of the vectors. For instance for a third order tensor $\mathcal{X} = u \circ v \circ w$ where $u \in \mathbb{R}^2, v \in \mathbb{R}^3$ and $w \in \mathbb{R}^2$, we can see that

$$\text{vec}(\mathcal{X}) = \begin{pmatrix} u_1v_1w_1 \\ u_2v_1w_1 \\ \vdots \\ u_1v_3w_2 \\ u_2v_3w_2 \end{pmatrix} = w \otimes v \otimes u$$

this result can be generalized easily for order N tensor $\mathcal{X} = a^{(1)} \circ \ldots \circ a^{(N)}$, i.e.

$$\text{vec}(\mathcal{X}) = \text{vec}(a^{(1)} \circ \ldots \circ a^{(N)}) = a^{(N)} \otimes \ldots \otimes a^{(1)}.$$

The modal unfoldings (matricizations) of rank one tensors have a useful structure as well, one can verify by a simple Kronecker multiplication that

$$X_{(1)} = \begin{pmatrix} u_1v_1w_1 & \ldots & u_1v_3w_2 \\ u_2v_1w_1 & \ldots & u_2v_3w_2 \end{pmatrix} = u \otimes (w \otimes v)^T,$$

$$X_{(2)} = \begin{pmatrix} u_1v_1w_1 & \ldots & u_2v_1w_2 \\ u_1v_2w_1 & \ldots & u_2v_2w_2 \\ u_1v_3w_1 & \ldots & u_2v_3w_2 \end{pmatrix} = v \otimes (w \otimes u)^T,$$

and

$$X_{(3)} = \begin{pmatrix} u_1v_1w_1 & \ldots & u_2v_3w_1 \\ u_1v_1w_2 & \ldots & u_2v_3w_2 \end{pmatrix} = w \otimes (v \otimes u)^T.$$

In general, if $a^{(n)} \in \mathbb{R}^{I_n}$ for $n = 1, \ldots N$, and

$$\mathcal{X} = a^{(1)} \circ \ldots \circ a^{(N)} \in \mathbb{R}^{I_1 \times \ldots \times I_N},$$

then its modal unfoldings (matricizations) are rank-1 matrices:

$$X_{(n)} = a^{(n)} \left(a^{(N)} \otimes \ldots \otimes a^{(n+1)} \otimes a^{(n-1)} \otimes \ldots \otimes a^{(1)} \right)^T. \tag{2.1}$$

The n-mode (vector) product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ with a vector $v \in \mathbb{R}^{I_n}$ produces a $N-1$ order tensor $\mathcal{X} \times_n v$ where its elements are defined as follows

$$(\mathcal{X} \times_n v)_{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_N} v_{i_n}.$$

Note that $\mathcal{X} \times_n v$ is a new tensor of size $I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N$. A tensor can be multiplied by several vectors in this way. For example assume that $v^{(n)} \in \mathbb{R}^{I_n}$ for $n = 1, \ldots, N$, then multiplying in all modes will produce a scalar:

$$\begin{aligned} \mathcal{X} \times_{n=1}^{N} v^{(n)} &= \mathcal{X} \times_1 v^{(1)} \times_2 \ldots \times_N v^{(N)} \\ &= \sum_{i_1=1}^{I_N} \ldots \sum_{i_N=1}^{I_N} x_{i_1 \ldots i_N} v_{i_1}^{(1)} \ldots v_{i_N}^{(N)}. \end{aligned}$$

The n-mode multiplication in every mode except mode n results in a vector of length I_n, i.e.

$$\mathcal{X} \times_{m \neq n}^{N} v^{(m)} = X_{(n)} v^{(-n)},$$

where

$$v^{(-n)} = v^{(N)} \otimes \ldots \otimes v^{(n+1)} \otimes \ldots \otimes v^{(1)}.$$

Similarly the multiplication in every mode except n and p results in a matrix of size $I_n \times I_p$.

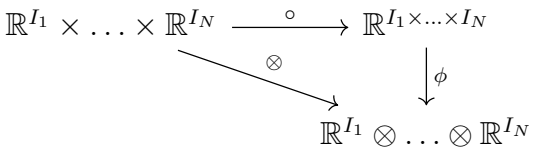

FIGURE 2.11. The universal property of the tensor product

2.3. Tensor as an Element of Tensor Product Spaces

Recall that in multilinear algebra, a tensor is simply an element in the tensor product of vector spaces [26]. The tensor product $\mathbb{R}^{I_1} \otimes \ldots \otimes \mathbb{R}^{I_N}$ with the multilinear map $\otimes$ always exists. One can easily check that the outer product defined in (2.1.4) is multilinear [44]. In particular the mapping

$$\circ : \mathbb{R}^{I_1} \times \ldots \times \mathbb{R}^{I_N} \to \mathbb{R}^{I_1 \times \ldots \times I_N},$$

is linear on each component. So by the universal property of the tensor product, there exists a unique linear mapping ϕ such that the diagram (2.11) commutes, note that since $\otimes$ generates the tensor product space $\mathbb{R}^{I_1} \otimes \ldots \otimes \mathbb{R}^{I_N}$, dimensions of $\mathbb{R}^{I_1 \times \ldots \times I_N}$ and $\mathbb{R}^{I_1} \otimes \ldots \otimes \mathbb{R}^{I_N}$ are equal, therefore ϕ is an isomorphism. If we consider the canonical basis of $\mathbb{R}^{I_1} \otimes \ldots \otimes \mathbb{R}^{I_N}$,

$$\{e_{i_1}^{(1)} \otimes \ldots \otimes e_{i_N}^{(N)} \,|\, 1 \leq i_j \leq I_j,\, j = 1, \ldots N\}$$

then ϕ can be described as

$$\phi\left(\sum_{i_1,\ldots,i_N} x_{i_1 \ldots i_N} e_{i_1}^{(1)} \otimes \ldots \otimes e_{i_N}^{(N)}\right) = \mathcal{X}.$$

Therefore there is a one to one correspondence between the elements of tensor product space $\mathbb{R}^{I_1} \otimes \ldots \otimes \mathbb{R}^{I_N}$ and the multi-way arrays in the space $\mathbb{R}^{I_1 \times \ldots \times I_N}$. As a vector space, the tensor space $\mathbb{R}^{I_1 \times \ldots \times I_N}$ can be equipped with scalar multiplication and addition:

$$[c\,\mathcal{X}]_{i_1 \ldots i_N} = c x_{i_1 \ldots i_N},$$

and

$$[\mathcal{X} + \mathcal{Y}]_{i_1 \ldots i_N} = x_{i_1 \ldots i_N} + y_{i_1 \ldots i_N}.$$

2.4. Tensor Rank

Any given tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ can always be decomposed as

$$\mathcal{X} = \sum_{r=1}^{R} a_r^{(1)} \circ \ldots \circ a_r^{(N)}. \tag{2.2}$$

The rank of a tensor denoted by rank($\mathcal{X}$) is the smallest positive integer R such that the above decomposition is exact [30, 31], where "exact" means that the equality holds in (2.2). A decomposition of a tensor in the form (2.2) where R =rank($\mathcal{X}$) is called the rank decomposition of $\mathcal{X}$. The definition of the rank of a tensor is analogous to the one in matrices, however there are many differences. One difference is that a tensor may have different ranks over $\mathbb{R}$ and $\mathbb{C}$. For example consider the third order tensor $\mathcal{X} \in \mathbb{R}^{2\times 2\times 2}$ with frontal slices

$$X(:,:,1) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X(:,:,2) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

has the rank decomposition over $\mathbb{R}$ with the vectors

$$a_1^{(1)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, a_2^{(1)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, a_3^{(1)} = \begin{pmatrix} 1 \\ -1 \end{pmatrix},$$

$$a_1^{(2)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, a_2^{(2)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, a_3^{(2)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

and

$$a_1^{(3)} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, a_2^{(3)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, a_3^{(3)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The proof that this given tensor is rank three over $\mathbb{R}$ can be found in [42]. On the other hand, the rank decomposition of $\mathcal{X}$ can be obtained over $\mathbb{C}$ by the following vectors:

$$a_1^{(1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -i \end{pmatrix}, a_2^{(1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ i \end{pmatrix},$$

$$a_1^{(2)} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ i \end{pmatrix}, a_2^{(2)} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -i \end{pmatrix},$$

and

$$a_1^{(3)} = \begin{pmatrix} 1 \\ i \end{pmatrix}, a_2^{(3)} = \begin{pmatrix} 1 \\ -i \end{pmatrix}.$$

lk For further discussion about this example see [71]. Another major difference between matrix and tensor rank is that unlike matrices there is no direct relation between the dimensions of a tensor and its rank. Recall that in matrix case, the relation $\text{rank}(X) \leq \min\{I, J\}$ is always valid. For example, consider a square matrix $X \in \mathbb{R}^{2\times 2}$ whose elements are drawn from the standard normal distribution (e.g. randn(2,2) in MATLAB) , it can be shown that with probability one the rank of such a matrix is equal to two. In contrast, a randomly generated third order tensor $\mathcal{X} \in \mathbb{R}^{2\times 2\times 2}$ may have different ranks over $\mathbb{R}$, In fact

$$\text{rank}(\mathcal{X}) = \begin{cases} 2, & \text{with probability } \pi/4 \\ 3, & \text{with probability } 1 - \pi/4. \end{cases}$$

However it has rank two over $\mathbb{C}$ with probability equal to one. Consider the space of all tensors $\mathcal{X} \in \mathbb{R}^{I_1\times\ldots\times I_N}$ with rank R over $\mathbb{R}$, we denote this space with $T_R^{I_1\times\ldots\times I_N}$, in particular

$$T_R^{I_1\times\ldots\times I_N} = \{\mathcal{X} \in \mathbb{R}^{I_1\times\ldots\times I_N} | \text{rank}(\mathcal{X}) = R\} \tag{2.3}$$

we say that R is a typical rank of $\mathcal{X} \in \mathbb{R}^{I_1\times\ldots\times I_N}$ if the measure of set $T_R^{I_1\times\ldots\times I_N}$ is not zero. For example, by the above discussion, a third order tensor $\mathcal{X} \in \mathbb{R}^{2\times 2\times 2}$ has two typical ranks two and three over $\mathbb{R}$ but only one typical rank over $\mathbb{C}$. When there is only one typical rank (that occurs with probability one then) we call it generic rank. Table (2.1) shows the typical rank of some third order tensors with different dimensions over $\mathbb{R}$.

Consider the third order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ where

$$\mathcal{X} = a \circ a \circ b + a \circ b \circ a + b \circ a \circ a$$

Size of Tensor	Typical rank	Citation
$2 \times 2 \times 2$	$2, 3$	[42]
$3 \times 3 \times 2$	$3, 4$	[41]
$5 \times 3 \times 3$	$5, 6$	[69]
$I \times J \times 2,\ I \geq 2J$	$2J$	[70]
$I \times J \times 2$	$I, I+1$	[70]
$I \times J \times K,\ I \geq JK$	JK	[68]

TABLE 2.1. Typical rank of third order tensors over $\mathbb{R}$

where $\|a\| = \|b\| = 1$ and $\langle a, b\rangle \neq 1$. This tensor has rank three [54], however it can be approximated by the following sequence of rank two tensors [36]:

$$\mathcal{X}_n = n\left(a + \frac{1}{n}b\right) \circ \left(a + \frac{1}{n}b\right) \circ \left(a + \frac{1}{n}b\right) - na \circ a \circ a$$

note that $\mathcal{X}_n \to \mathcal{X}$ as $n \to \infty$. Although $\mathcal{X}$ has rank three, it can be well approximated by a sequence of rank two tensors. In such cases we say $\mathcal{X}$ has the border rank two. This shows if we let R to be equal to border rank of $\mathcal{X}$, then the following optimization problem is ill-posed

$$\min_{a_r, b_r, c_r} \|\mathcal{X} - \sum_{r=1}^{2} a_r \circ b_r \circ c_r\|_F^2,$$

meaning that the minimum does not exists. In other words the set $T_R^{I_1 \times \ldots \times I_N}$ defined in (2.3) is not closed under the induced topology of Frobenius norm. This example illustrates a special case in tensor rank which is called degeneracy. A tensor is degenerate if it can be approximated arbitrarily small by a sequence of lower rank tensors. As this example shows, the factors become nearly proportional to each other and the norm of some factors approaches to infinity as $n \to \infty$. Figure (2.12) shows the degeneracy problem of estimating a rank three tensor $\mathcal{Y}$ by a sequence of rank two tensors. In this example, a sequence $\{\mathcal{X}_k\}$ of rank two tensors make a better estimation of $\mathcal{Y}$ as $k \to \infty$. The best approximation of $\mathcal{Y}$ lies in the border of rank

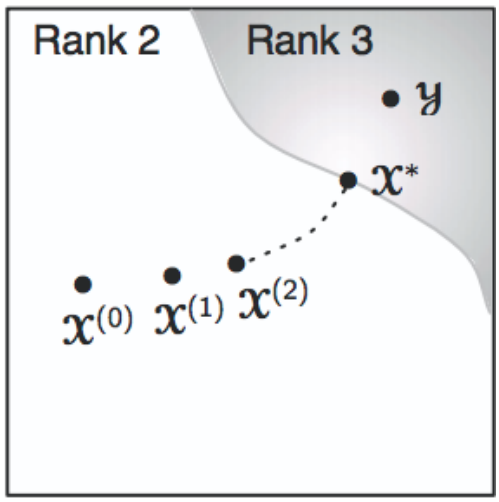

FIGURE 2.12. A sequence of rank two tensors converging to a rank three tensor [36].

two and rank three tensors. However the limit of the sequence does not belong to the space of rank two tensors due the lack of closedness of the space of rank two tensors. For third order tensors, the rank may be as large as $\min(IJ, JK, IK)$. In fact this is the maximum attainable rank for the third order tensors. This upper bound, although not very tight, is important because it states that the tensor rank is finite. It is also useful to have the lower bound for the third order tensors. we have

$$\max(I, J, K) \leq \text{rank}(\mathcal{X}) \leq \min(IJ, JK, IK). \tag{2.4}$$

In the case where a best low approximation does not exist, we consider the concept of border rank. The border rank of a tensor is defined as the minimum number of rank one tensors that approximate a given tensor with arbitrary small residual error. Mathematically, given $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, the rank of $\mathcal{X}$ is defined as

$$\text{rank}_B(\mathcal{X}) = \min\{r | \text{for } \epsilon > 0,\ \exists\ \mathcal{E}\ , \|\mathcal{E}\| < \epsilon\ \text{rank}(\mathcal{X} + \mathcal{E}) = r\}, \tag{2.5}$$

A trivial inequality holds between the border rank and the rank of a tensor:

$$\operatorname{rank}_B(\mathcal{X}) \leq \operatorname{rank}(\mathcal{X}).$$

CHAPTER 3

Optimization and Proximal Algorithms

3.1. Introduction

Recall that the CP problem can be formulated as an optimization problem. In this chapter we review some basic methods and algorithms which we will use in the next chapters. Most of the algorithms we use for tensor decomposition and rank approximation are of the type of block coordinate descent (BCD). Block coordinate descent methods of Gauss-Seidel type minimizes the objective function cyclically over each block while fixing the remaining blocks at their last value. The optimization problems we focus on have the following form

$$\min_x F(x_1, \ldots, x_s) = f(x_1 \ldots, x_s) + \sum_{i=1}^{s} r_i(x_i), \tag{3.1}$$

where the variable x has s blocks $x_1, \ldots, x_s$. The BCD updates for the problem (3.1) can have different forms:

(1) The primal update minimizes the objective function over a specific block:

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} F(x_1^{k+1}, \ldots x_{i-1}^{k+1}, x_i, x_{i+1}^{k}, \ldots, x_s^{k}). \tag{3.2}$$

Alternating least squares (ALS) is an example of this type of update. The primal update is the most used form in BCD. However, the convergence of the method relies on the convexity and differentiability of F. When F is not convex or differentiable, it may cycle and stagnate [58] or can get stuck in a nonstationary point [5].

(2) The proximal update minimizes the objective function plus a proximal term over a specific block at each iteration:

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} F(x_1^{k+1}, \ldots x_{i-1}^{k+1}, x_i, x_{i+1}^{k}, \ldots, x_s^{k}) + \frac{L_i}{2}\|x_i - x_i^k\|_2^2. \tag{3.3}$$

This update was also applied on ALS in order to reduce the swamp [49]. The convergence of the method has been discussed in [4].

(3) The proximal linear update minimizes L_f, the linearized f, with an additional proximal term over each block:

$$x_i^{k+1} \in \underset{x_i}{\operatorname{argmin}}\, L_f(x_i) + r_i(x_i) + \frac{L_i}{2}\|x_i - x_i^k\|_2^2, \tag{3.4}$$

where

$$L_f(x_i) = f(x_i^k) + \langle x_i - x_i^k, \nabla_i f(x_i^k)\rangle.$$

The proximal linear update is new but very similar to the block coordinate gradient descent (BCGD) which was discussed in [67]. This update is used in chapter 5 in order to solve the rank approximation problem of a tensor.

In the next section, we discuss basic optimization techniques. Then, we continue by reviewing some basic proximal algorithms which will be used later on chapter five.

3.2. Gradient Methods

In this section we review the standard gradient methods for minimizing a given objective function f. We start with the following definition:

DEFINITION 3.2.1. *[9] A vector x^* is called a local minimum of f, if there exists an $\epsilon > 0$ such that*

$$f(x^*) \leq f(x), \qquad \textit{when} \quad \|x^* - x\| < \epsilon.$$

It is called a global minimum of f if

$$f(x^*) \leq f(x) \qquad x \in \mathbb{R}^n.$$

PROPOSITION 3.1. *(First order necessary condition) Let x^* be a local minimum of $f : \mathbb{R}^n \to \mathbb{R}$, and assume that f is continuously differentiable, then*

$$\nabla f(x^*) = 0.$$

Given the initial point $x_0 \in \mathbb{R}^n$, an iterative descent method generates a sequence of points $\{x^k\}$ such that

$$f(x^{k+1}) < f(x^k), \quad k = 0, 1, \ldots$$

This process successively improves the current solution estimate and we hope to reach the minimum of f. One way to obtain the vector x^{k+1} from x^k is to search along the half line of vectors of the form

$$x^{k+1} = x^k + \alpha^k d^k,$$

where $\alpha^k \geq 0$ represents the step-size that we take at each iteration and $d^k \in \mathbb{R}^n$ is the search direction. From the first order Taylor series expansion around x^k, we have

$$\begin{aligned} f(x^{k+1}) &= f(x^k) + \nabla f(x^k)^T (x^{k+1} - x^k) + o(\|x^{k+1} - x^k\|_2) \\ &= f(x^k) + \nabla f(x^k)^T (x^k + \alpha^k d^k - x^k) + o(\alpha^k) \\ &= f(x^k) + \alpha^k \nabla f(x^k)^T d^k + o(\alpha^k). \end{aligned}$$

Since the term $\alpha^k \nabla f(x^k)^T d^k$ dominates $o(\alpha^k)$ near zero, we must have $\nabla f(x^k)^T d^k < 0$ in order to have a descent iteration. This is equivalent to say that the search direction must make an obtuse angle with the gradient vector at point x^k. A very natural candidate for search direction is $d^k = -\nabla f(x^k)$ whenever $\nabla f(x^k) \neq 0$. We call the algorithms of this type gradient methods. In general, the gradient methods have the following form

$$x^{k+1} = x^k - \alpha^k D^k \nabla f(x^k),$$

where D^k is a positive definite symmetric matrix. Note that

$$\nabla f(x^k)^T D^k \nabla f(x^k) > 0$$

due to the positive definiteness of D^k.

EXAMPLE 3.2.1. *The simplest choice of D^k is $D^k = I$, where I represents the identity matrix. This methods is called the steepest descent. Despite the simplicity of the method it often leads to a slow convergence.*

EXAMPLE 3.2.2. *In Newton method, we let $D^k = \left(\nabla^2 f(x^k)\right)^{-1}$ at each iteration, provided that the Hessian matrix $\nabla^2 f(x^k)$ is positive definite at each iteration.*

The Newton method can be interpreted as minimizing the quadratic model of the objective function around the current point x^k. Let m^k represent the quadratic model of the objective function f around x^k, i.e.

$$m^k(x) = f(x^k) + \nabla f(x^k)^T(x - x^k) + \frac{1}{2}(x - x^k)^T \nabla^2 f(x^k)(x - x^k)$$

then we find the minimum of m^k by equating its gradient to zero,

$$\nabla m^k(x) = \nabla f(x^k) + \nabla^2 f(x^k)(x - x^k) = 0$$

and we obtain the x^{k+1} as the minimum of $m^k(x)$:

$$x^{k+1} = x^k - \left(\nabla^2 f(x^k)\right)^{-1} \nabla f(x^k).$$

The general form of the Newton method can be expressed as

$$x^{k+1} = x^k - \alpha^k \left(\nabla^2 f(x^k)\right)^{-1} \nabla f(x^k),$$

with the step size $\alpha^k > 0$. Unlike the steepest descent method, Newton method converges very fast asymptotically.

PROPOSITION 3.2. *[52] Suppose that f is twice differentiable and that the Hessian $\nabla^2 f(x)$ is Lipschitz continuous near the minimizer x^* and is positive definite, then the Newton method has the following properties:*

(1) if the initial guess x^0 is sufficiently close to x^*, the sequence $\{x^k\}$ converges to x^*;

(2) the rate of convergence of $\{x^k\}$ is quadratic; and

(3) the sequence of gradient norms $\{\|\nabla f(x^k)\|_2\}$ converges to zero quadratically.

3.3. Least Squares Problem

3.3.1. Linear Least Squares. Consider the overdetermined linear system

$$Ax = b, \tag{3.5}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $m \geq n$. The vector x^* is called the least-squares solution of (3.5) if it solves the following minimization problem

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax - b\|_2^2. \tag{3.6}$$

When A is full column rank, i.e. $\text{rank}(A) = n$, then it can be shown that the optimization problem (3.6) has a unique solution. In particular the unique solution of (3.6) can be expressed by the following normal equation:

$$A^TAx = A^Tb. \tag{3.7}$$

Note that the above normal equation is obtained by the first order optimality condition. If we let f be the objective function in the equation (3.6), then we have

$$\nabla f(x) = A^T(Ax - b), \quad \nabla^2 f(x) = A^TA.$$

Thus, when A is full rank the Hessian matrix of f is positive definite which means f is strictly convex so it must have a unique minimizer. The full rankness of A implies invertibility of A^TA. Hence the unique solution of (3.6) has the following closed form

$$x^* = (A^TA)^{-1}A^Tb.$$

When $\text{rank}(A) = m$, A^TA is not invertible so the optimization problem (3.6) has infinitely many solutions. However, it can be shown that the vector

$$x^* = A^T(AA^T)^{-1}b,$$

is the only solution among all the solutions with minimum ℓ_2 norm. When A has many rows (m is large), calculating $(AA^T)^{-1}$ is very expensive. The Kaczmarz's algorithm [34] is an iterative technique to find the solution of (3.6) without calculating $(AA^T)^{-1}$. The global convergence of Kaczmarz's algorithm was discussed in [56]. In general,

$A^\dagger b$ provides the solution with the minimum norm, so we have the following theorem [14]:

THEOREM 3.1. *Consider the least squares problem stated in (3.6). If rank$(A) = r$, then the vector $x^* = A^\dagger b$ minimizes the objective function on $\mathbb{R}^n$. Furthermore among all vectors in $\mathbb{R}^n$ that minimizes (3.6), x^* is the unique vector with minimal norm.*

3.3.2. Nonlinear Least Squares. Let $f : \mathbb{R}^n \to \mathbb{R}^m$ $(m \geq n)$ be a smooth function. We want to find a vector x^* such that $\|f\|_2^2$ is minimized, In other words we are interested to solve the following minimization problem

$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|f(x)\|_2^2. \tag{3.8}$$

Let us denote the objective function in (3.8) by F, then

$$F(x) = \frac{1}{2}\sum_{i=1}^{m}(f_i(x))^2 = \frac{1}{2}\|f(x)\|_2^2.$$

Provided that f has continuous second partial derivatives, we can calculate the gradient and Hessian of F:

$$\frac{\partial F}{\partial x_j}(x) = \sum_{i=1}^{m} f_i(x)\frac{\partial f_i}{\partial x_j}(x), \tag{3.9}$$

therefore the gradient vector of F is

$$\nabla F(x) = J^T(x)f(x),$$

where $J(x)$ represents the Jacobian of f, i.e.

$$(J(x))_{ij} = \frac{\partial f_i}{\partial x_j}(x).$$

We can calculate the Hessian of F by the equation (3.9):

$$\frac{\partial^2 F}{\partial x_j \partial x_k}(x) = \sum_{i=1}^{m}\left(\frac{\partial f_i}{\partial x_j}(x)\frac{\partial f_i}{\partial x_k}(x) + f_i(x)\frac{\partial^2 f_i}{\partial x_j \partial x_k}(x)\right) \tag{3.10}$$

which shows that

$$\nabla^2 F(x) = J^T(x)J(x) + \sum_{i=1}^{m} f_i(x)\nabla^2 f_i(x).$$

A simple case is when f has the linear form

$$f(x) = b - Ax,$$

where the vector $b \in \mathbb{R}^m$ and matrix $A \in \mathbb{R}^{m \times n}$. In this case $J(x) = -A$ for all x and we have that

$$F'(x) = -A^T(b - Ax).$$

In MATLAB, the command "$A \setminus b$" returns the least squares solution computed via orthogonal transformation. In the next two sections we review two of the most effective algorithms for solving the non linear least squares problems.

3.3.3. The Gauss-Newton Method. The Gauss-Newton method is a very effective method for solving nonlinear least squares problem. It is based on the first derivative of component functions f_i, $i = 1, \ldots m$. In special cases it can provide quadratic rate of convergence as the Newton method does for optimization problems [24]. Consider the linear approximation of f near a vector x (for small $\|h\|$), i.e.

$$f(x + h) \approx l(h) = f(x) + J(x)h, \tag{3.11}$$

Then the objective function F can be approximated by l as follow

$$\begin{aligned} F(x + h) \approx L(h) &= \frac{1}{2}\|l(h)\|_2^2 \\ &= \frac{1}{2} l(h)^T l(h) \\ &= \frac{1}{2} f(x)^T f(x) + h^T J^T(x) f(x) + \frac{1}{2} h^T J^T(x) J(x) h \\ &= F(x) + h^T J^T(x) f(x) + \frac{1}{2} h^T J^T(x) J(x) h \end{aligned}$$

Note that $\nabla L(h) = J^T f + J^T J h$ and $\nabla^2 L(h) = J^T J$, so h_{GN} can be obtained by solving the following normal equation

$$(J^T J) h_{GN} = -J^T f. \tag{3.12}$$

If the Jacobian matrix is full rank then J^TJ is positive definite and h_{GN} is determined uniquely at each iteration. This is a descent direction for F because

$$h_{GN}^T \nabla F(x) = h_{GN}(J^T f) = -h_{GN}^T (J^T J) h_{GN} < 0$$

due to the positive definiteness of J^TJ.

3.3.4. The Levenberg-Marquardt Method. Levenberg-Marquardt algorithm was first published in 1944 by Kenneth Levenberg [46] and was rediscovered in 1963 by Donald Marquardt [48]. The step h_{LM} is obtained by the following modification to (3.12), i.e.

$$(J^TJ + \mu I)h_{LM} = -J^T f, \quad \mu > 0, \tag{3.13}$$

where μ is the damping parameter. The advantages of damping parameter is the following:

(1) For all $\mu > 0$, the coefficient matrix is positive definite. This ensures that h_{LM} is a descent direction. Also h_{LM} is uniquely determined at each iteration.

(2) For large values of μ, we can neglect the term J^TJ, so the method reduces to gradient descent method:

$$h_{LM} \approx -\frac{1}{\mu} J^T f,$$

which is "good" if the iterates are far from the solution.

(3) For small values of μ, the method is similar to Gauss-Newton, i.e. $h_{LM} \approx h_{GN}$. This is "good" if we are close enough to the solution. Also the quadratic rate of convergence can be obtained from it.

In order to update the damping parameter μ we define the gain ratio

$$\sigma = \frac{F(x) - F(x + h_{LM})}{L(0) - L(h_{LM})},$$

where the denominator represents the predicted gain by the linear model L,

$$\begin{aligned} L(0) - L(h_{LM}) &= -h_{LM}^T J^T f - \frac{1}{2} h_{LM}^T J^T J h_{LM} \\ &= -\frac{1}{2} h_{LM}^T \left(2J^T f + J^T J h_{LM}\right) \\ &= \frac{1}{2} h_{LM}^T \left(\mu h_{LM} - J^T f\right) > 0. \end{aligned}$$

If σ is large, then the linear model L is a good approximation of F, so we are interested in taking a larger step by reducing μ. On the other hand a small value of σ indicates that L is a poor approximation of F, therefore we increase μ in order to get closer to gradient descent method which takes shorter step length. Algorithm (2) describes the Levenberg-Marquardt method. The stopping criterion for the algorithm is when

Algorithm 1 Levenberg-Marquardt Method

Initialize $x = x_0$, $\nu = 2$, $A = J(x)^T J(x)$, $g = J(x)^T f(x)$

while stopping criteria not met **do**

solve $(A + \mu I)h_{LM} = -g$

$x_{new} = x + h_{LM}$

$\sigma = (F(x) - F(x_{new}))/(L(0) - L(h_{LM})$

if $\sigma > 0$ **then**

$x = x_{new}$, $A = J(x)^T J(X)$, $g = J(x)^T f(x)$

$\mu = \mu/3$, $\nu = 2$

else

$\mu = mu * \nu$, $\nu = 2 * \nu$

end if

end while

we are close to a stationary point, i.e. $F'(x^*) = g(x^*) = 0$, so we can impose the condition

$$\|g\|_\infty \leq \epsilon,$$

where ϵ is very small, positive number. Another stopping criterion is when we get a very small change in x, i.e.

$$\|x_{new} - x\| \leq \epsilon(\|x\| + \epsilon),$$

finally, like all iterative processes we need to put a maximum number of iterations to avoid falling into an infinite loop. There are other methods which effectively solve the nonlinear least squares problem such as Dog Leg method, quasi Newton methods, etc [47]. However, we only use the Levenberg-Marquardt Method in the following chapters therefore we do not state them here.

3.4. Proximal Operator

3.4.1. Proximal Mapping. In this section, we introduce the proximal operator for a closed proper convex function and review the basic properties of it. We follow the notations used in [53]. The proximal operator plays a key role in the analysis of the convergence of specific algorithms which will be introduced in the future chapters.

DEFINITION 3.4.1. *[53] Let $f : \mathbb{R}^n \to \mathbb{R} \cup +\infty$ be a closed proper convex function, the proximal operator of f with parameter $\lambda > 0$, $prox_{\lambda f} : \mathbb{R}^n \to \mathbb{R}$, is defined by*

$$prox_{\lambda f}(y) = \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{1}{2\lambda}\|x - y\|_2^2 \right),$$

where $\|.\|_2^2$ represents the usual Euclidean norm.

Note that the function minimized on the righthand side is strongly convex so it has a unique minimizer for every $y \in \mathbb{R}^n$. The convex assumption in the definition above can be removed but this will result in having a set-valued operator, i.e. the minimizer of the function on the righthand side of the equation is not unique. In this case the proximal operator still remains well-defined:

PROPOSITION 3.3. *[60] Let $f : \mathbb{R}^n \to \mathbb{R} \cup \infty$ be a proper closed function with $\inf f > -\infty$. Then for every $\lambda > 0$ the set $prox_{\lambda f}$ is nonempty and compact.*

EXAMPLE 3.4.1. *The indicator function δ_C of a closed nonempty convex set $C \subset \mathbb{R}^n$ is defined as*

$$\delta_C(x) = \begin{cases} 0 & x \in C, \\ +\infty & x \notin C. \end{cases}$$

The proximal operator of δ_C reduces to the Euclidean projection onto C, i.e.

$$\begin{aligned} prox_{\lambda\delta_C}(y) &= \operatorname*{argmin}_x \left(\delta_C(x) + \frac{1}{2\lambda}\|y - x\|_2^2 \right) \\ &= \operatorname*{argmin}_{x \in C} \|y - x\|_2 = \Pi_C(y) \end{aligned}$$

EXAMPLE 3.4.2. *Let $f(x) = |x|$, then we have that*

$$prox_{\lambda f}(y) = \begin{cases} y - \lambda & v \geq \lambda \\ 0 & |y| \leq \lambda \\ y + \lambda & y \leq -\lambda \end{cases}$$

This operation is called soft thresholding.

If $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is fully separable, i.e.

$$f(x) = \sum_{i=1}^{n} f_i(x),$$

then $(\text{prox}_{\lambda f}(y))_i = \text{prox}_{\lambda f_i}(y_i)$, this means the proximal operator of fully separable function reduces to evaluating the proximal operator of scalar function.

EXAMPLE 3.4.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be the ℓ_1 norm, i.e.*

$$f(x) = \|x\|_1 = \sum_{i=1}^{n} |x_i|,$$

then the proximal operator of f with parameter $\lambda > 0$ becomes

$$(\text{prox}_{\lambda f}(y))_i = \begin{cases} y_i - \lambda & y_i \geq \lambda \\ 0 & |y_i| \leq \lambda \\ y_i + \lambda & y_i \leq -\lambda. \end{cases}$$

PROPOSITION 3.4. *[53] (Basic properties of proximal operator) Let $\lambda, \gamma > 0$ be positive constants,*

(1) If $f(x) = \gamma g(x) + C$, then

$$prox_{\lambda f}(y) = prox_{\gamma\lambda g}(y).$$

(2) If $f(x) = g(x) + a^T x + C$, then

$$prox_{\lambda f}(y) = prox_{\lambda g}(y - \lambda a).$$

(3) If $f(x) = g(Qx)$, where Q is an orthogonal matrix, then

$$prox_{\lambda f}(y) = Q^T prox_{\lambda g}(Qy).$$

DEFINITION 3.4.2. *[6] The infimal convolution of closed proper convex functions f and g, denoted by $f\square g$, is defined as*

$$(f\square g)(y) = \inf_x (f(x) + g(y - x)).$$

Given $\lambda > 0$, the *Moreau envelope* of f with parameter λ, denoted by $M_{\lambda f}$ is the infimal convolution of λf and a scaled ℓ_2 norm, i.e.

$$M_{\lambda f}(y) = (\lambda f)\square \left(\frac{1}{2}\|.\|_2^2\right)(y) = \inf_x \left(f(x) + \frac{1}{2\lambda}\|x - y\|_2^2\right).$$

There is a close relation between the proximal operator of a function and the Moreau envelope of it, prox_f returns the (unique) minimizer of M_f,

$$M_f(y) = f(\text{prox}_f(y)) + \frac{1}{2}\|y - \text{prox}_f(y)\|_2^2$$

It can be shown that M_f is differentiable [6] and the derivative of M_f is given by

$$\nabla M_{\lambda f} = \lambda^{-1}(I - \text{prox}_{\lambda f}). \tag{3.14}$$

Furthermore, $\nabla M_{\lambda f}$ is Lipschitz continuous, with constant λ^{-1}, i.e.

$$\|\nabla M_{\lambda f}(y_1) - \nabla M_{\lambda f}(y_2)\|_2 \leq \lambda^{-1}\|y_1 - y_2\|_2,$$

for any $y_1, y_2 \in \mathbb{R}^n$.

EXAMPLE 3.4.4. *Let C be a closed convex set in $\mathbb{R}^n$. The distance function d_C is defined as*

$$d_C(x) = \inf_{y \in C} \|x - y\|.$$

This can be expressed as the infimal convolution of δ_C and $(1/2)\|.\|_2^2$, i.e.

$$d_C(x) = \delta_C \square \frac{1}{2}\|.\|_2^2.$$

3.4.2. Proximal Algorithms. In this subsection we review some basic proximal algorithms for solving convex and non-convex optimization problems. Consider the problem

$$\min f(x) + g(x) \tag{3.15}$$

where f and g are closed proper convex functions and f is differentiable. The proximal gradient method is

$$x^{k+1} = \text{prox}_{\lambda^k g}(x^k - \lambda^k \nabla f(x^k))$$

where $\lambda^k > 0$ is a step size. It can be shown [7] that the the proximal gradient method converges with rate $O(1/k)$ when ∇f is Lipschitz continuous with constant L and $\lambda^k = \lambda \in (1, 1/L]$. It is clear that when $g(x) = 0$, the proximal gradient method reduces to the standard gradient descent method. When $g(x) = \delta_C(x)$, the proximal gradient method is called the projected gradient method [9]. The accelerated version of proximal gradient method includes the extrapolation term in the algorithm:

$$y^{k+1} = x^k + \omega^k(x^k - x^{k-1})$$

$$x^{k+1} = \text{prox}_{\lambda^k g}(y^{k+1} - \lambda^k \nabla f(y^{k+1}))$$

where $\omega^k \in (0, 1)$ is the extrapolation parameter. There are several ways to determine ω^k at each iteration. A very simple choice [76] takes

$$\omega^k = \frac{k}{k+3}.$$

For more information about accelerated proximal gradient methods and analysis of convergence of the method see [8].

Algorithm 2 Proximal gradient method

Initialize x^0, $\beta \in (0,1)$,

while stopping criteria not met **do**

$x^{k+1} = \text{prox}_{\lambda g}(x^k - \lambda \nabla f(x^k))$

$\lambda = \beta\lambda$

end while

The alternating direction method of multipliers (ADMM) is another method for solving problems of the form (3.15). It is also knows as Douglas-Rachford slitting. For the iteration counter k we have

$$
\begin{aligned}
x^{k+1} &= \text{prox}_{\lambda f}(z^k - u^k) \\
z^{k+1} &= \text{prox}_{\lambda g}(x^{k+1} + u^k) \qquad (3.16) \\
u^{k+1} &= u^k + x^{k+1} - z^{k+1}.
\end{aligned}
$$

For the convergence condition of this method see [12]. The advantage of ADMM is that the terms in the objective function (3.15) are handled separately and the functions are accessed through the proximal operators. ADMM is quite effective when computing the proximal operators of f and g is simple but evaluating the proximal of $f + g$ is not easy.

When $g = \delta_C$ is the indicator function of a closed convex set C, the proximal operator of g is the projection onto C. In this case, ADMM reduces to a method of minimizing f over C which only uses the proximal operator of f. ADMM can be interpreted as an augmented Lagrangian method. Note that the objective function in (3.15) can be rewritten as

$$
\min f(x) + g(z), \quad s.t. \quad x - z = 0. \qquad (3.17)
$$

The augmented Lagrangian associated with the problem (3.17) is

$$
L_\lambda(x, z, y) = f(x) + g(z) + y^T(x - z) + \frac{\lambda}{2}\|x - z\|_2^2,
$$

where $\lambda > 0$ and y is the dual variable. The ADMM method can then be expressed

$$x^{k+1} = \underset{x}{\operatorname{argmin}}\, L_\lambda(x, z^k, y^k)$$
$$z^{k+1} = \underset{z}{\operatorname{argmin}}\, L_\lambda(x^{k+1}, z, y^k)$$
$$y^{k+1} = y^k + \lambda(x^{k+1} - z^{k+1}).$$

In each step, L_λ is minimized over only one variable while it uses the most recent value of the other primal and dual variable. There are a number of widely used proximal algorithms, see, for example [12] and [53]. However, we do not use them in the future chapters so we do not state them here.

CHAPTER 4

Basic Tensor Decomposition

4.1. CANDECOMP/PARAFAC Decomposition

The main idea of tensor decomposition is to break it down to a group of simple structured tensors. Hitchcock [30], [31] introduced the idea of the polyadic decomposition of a tensor by breaking a tensor into the sum of a finite number of rank one tensors in 1927. Carroll and Chang [15] introduced CANDECOMP in 1970. Harshman [29] introduced PARAFAC in the context of principle component analysis. Currently, this decomposition is called the CANDECOMP/PARAFAC (CP) decomposition. The CP decomposition factorizes a tensor into a sum of rank one tensors. Given $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ and $R \in \mathbb{N}$, we would like to write the tensor in the following form

$$\mathcal{X} \approx \sum_{r=1}^{R} a_r^{(1)} \circ \ldots \circ a_r^{(N)}, \tag{4.1}$$

Equivalently, when we take the scalar weights for each rank one tensor component, it becomes

$$\mathcal{X} \approx \sum_{r=1}^{R} \alpha_r a_r^{(1)} \circ \ldots \circ a_r^{(N)}.$$

For each $1 \leq n \leq N$, we define the factor matrix $A^{(n)}$ as

$$A^{(n)} = (a_1^{(n)} \ldots a_R^{(n)}).$$

It is clear that $A^{(n)}$ is an $I_n \times R$ matrix for $1 \leq n \leq N$. If R is the actual rank of $\mathcal{X}$, the CP decomposition is called the rank decomposition of $\mathcal{X}$ and we have the equality instead of approximation in equation (4.10):

$$\mathcal{X} = \sum_{r=1}^{R} a_r^{(1)} \circ \ldots \circ a_r^{(N)}. \tag{4.2}$$

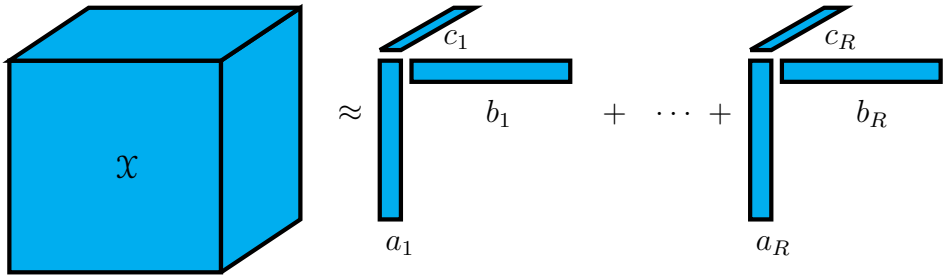

FIGURE 4.1. CP decomposition of a cubic tensor $\mathcal{X}$, with factor matrices $A = [a_1 \dots a_R]$, $B = [b_1 \dots b_R]$ and $C = [c_1 \dots c_R]$.

EXAMPLE 4.1.1. *Let $\mathcal{X} \in \mathbb{R}^{2\times 2\times 2}$ with frontal slabs*

$$X(:,:,1) = \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix}, \quad X(:,:,2) = \begin{pmatrix} 9 & 4 \\ 5 & 2 \end{pmatrix},$$

then $\mathcal{X}$ has the following rank decomposition

$$\mathcal{X} = a_1 \circ b_1 \circ c_1 + a_2 \circ b_2 \circ c_2,$$

with three factor matrices

$$A = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}.$$

Figure (4.1) illustrates the CP decomposition of a third order tensor.

4.1.1. Uniqueness of CP. For simplicity, let us focus on the third order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$. Assume that the CP decomposition of $\mathcal{X}$ is given by the following three factor matrices

$$A = \begin{pmatrix} a_1 \dots a_R \end{pmatrix}, \quad B = \begin{pmatrix} b_1 \dots b_R \end{pmatrix}, \quad C = \begin{pmatrix} c_1 \dots c_R \end{pmatrix}. \tag{4.3}$$

We adopt the following shorthand notation to represent the right hand side of the equation (4.2)

$$[A^{(1)}, \dots, A^{(N)}] = \sum_{r=1}^{R} a_r^{(1)} \circ \dots \circ a_r^{(N)}. \tag{4.4}$$

CP is unchanged by scaling; it means for a constant $c \neq 0$, we have that

$$[A, B, C] = [cA, c^{-1}B, C] = [cA, B, c^{-1}C] = [A, cB, c^{-1}C].$$

In general for any three non-zero scalars a, b, c where $abc = 1$ we have

$$[A, B, C] = [aA, bB, cC].$$

This shows that the CP decomposition is not unique. Furthermore if Π is a permutation matrix then

$$[A, B, C] = [A\Pi, B\Pi, C\Pi]$$

Assume that Π_{12} is a permutation matrix which exchanges the first two columns of a given matrix, then

$$[A, B, C] = [A\Pi_{12}, B\Pi_{12}, C\Pi_{12}] = \sum_{r=1}^{R} a_r \circ b_r \circ c_r.$$

So the uniqueness of CP problem is restricted to scaling and permutation of factor matrices. In addition Kruskal in 1977 [43] showed that the uniqueness of CP decomposition under some specific rank of factor matrices conditions.

DEFINITION 4.1.1. *Given a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, we say that its CP is essentially unique if the factor matrices A, B, C are uniquely determined (up to scaling and permutation).*

DEFINITION 4.1.2. *The Kruskal rank k_A of an $I \times R$ matrix A is defined to be the largest positive integer k such that any k columns of A are linearly independent.*

It is clear that $k_A \leq \text{rank}(A) \leq \min\{I, R\}$ for example let

$$A = \begin{pmatrix} 1 & 0 \\ 2 & 0 \end{pmatrix}$$

then $k_A = 0$ but $\text{rank}(A) = 1$.

THEOREM 4.1. *[43] Assume that $\mathcal{X} = [A, B, C]$, where $A \in \mathbb{R}^{I\times R}$, $B \in \mathbb{R}^{J\times R}$ and $C \in \mathbb{R}^{K\times R}$. If $k_A + k_B + k_C \geq 2R + 2$, then rank$(\mathcal{X}) = R$ and the CP decomposition of $\mathcal{X}$ is essentially unique.*

The result of this theorem was generalized later in 2000 [63].

THEOREM 4.2. *Given $\mathcal{X} = [A^{(1)}, \ldots, A^{(N)}]$. If $\sum_{n=1}^{N} k_{A^{(n)}} \geq 2R + N - 1$, then the CP decomposition of $\mathcal{X}$ is essentially unique.*

4.2. HOSVD/Tucker decomposition

Any matrix $A_{I\times J}$ with rank$(A) = R$ can be decomposed via SVD as $A = U\Sigma V^T$, where $U_{I\times I}$ and $V_{J\times J}$ are orthogonal matrices and $\Sigma_{I\times J}$ is a diagonal matrix with positive entries. Therefore we can write

$$A = \sum_{r=1}^{R} \Sigma(r, r) U_r V_r^T.$$

The goal of higher order SVD (HOSVD) is to generalize SVD to tensors. The Tucker decomposition was first introduced in 1963 [73], It decomposes a tensor into a core tensor multiplied by a matrix on each mode. For example if $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ is a third order tensor, then

$$\mathcal{X} \approx \mathcal{S} \times_1 A \times_2 B \times_3 C = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} s_{pqr} a_p \circ b_q \circ c_r = [\mathcal{S}; A, B, C], \tag{4.5}$$

where $A \in \mathbb{R}^{I\times P}$, $B \in \mathbb{R}^{J\times Q}$, and $C \in \mathbb{R}^{K\times R}$ are the orthogonal factor matrices. The core tensor $\mathcal{S} \in \mathbb{R}^{P\times Q\times R}$ indicates the level of interaction between the components. The elementwise formulation of (4.5) can be expressed as

$$x_{ijk} \approx \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} s_{pqr} a_{ip} b_{jq} c_{kr}, \quad i = 1, \ldots, I, j = 1, \ldots, J, k = 1, \ldots K. \tag{4.6}$$

Here P, Q, R are the number of columns in the factor matrices A, B, C respectively. It can be seen that the CP decomposition is a particular case of Tucker decomposition

when $P = Q = R$ and the core tensor $\mathcal{S}$ is diagonal. The Tucker model can be generalized to N dimensional tensors

$$\mathcal{X} = \mathcal{S} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \ldots \times_N A^{(N)} = [\mathcal{S}; A^{(1)}, A^{(2)}, \ldots, A^{(N)}], \tag{4.7}$$

or, elementwise, as

$$x_{i_1 i_2 \ldots i_N} = \sum_{r_1=1}^{R_1} \ldots \sum_{r_N=1}^{R_N} s_{r_1 r_2 \ldots r_N} a^{(1)}_{i_1 r_1} a^{(2)}_{i_2 r_2} \ldots a^{(N)}_{i_N r_N}, \quad i_n = 1, \ldots, I_N. \tag{4.8}$$

Equation (4.7) can also be expressed in the matrix form using tensor matricization. The result will be N matrix equations:

$$X_{(n)} = A^{(n)} S_{(n)} (A^{(N)} \otimes \ldots \otimes A^{(n+1)} \otimes A^{(n-1)} \otimes \ldots A^{(1)})^T, \tag{4.9}$$

for $n = 1, \ldots, N$. For a given N th-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, we define the n-rank of it to be the column rank of $X_{(n)}$ which we denote as $\text{rank}_n(\mathcal{X})$. If $R_n = \text{rank}_n(\mathcal{X})$, then we say that $\mathcal{X}$ is a rank-$(R_1, \ldots, R_N)$ tensor. It is clear that $R_n \leq I_n$ for each $n = 1, \ldots, N$.

THEOREM 4.3. *[25] (HOSVD) If* $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ *and*

$$X_{(n)} = U_n \Sigma_n V_n^T$$

are the SDVs of its modal unfoldings, then it HOSVD is given by

$$\mathcal{X} = \mathcal{S} \times_1 U_1 \times_2 \ldots \times_N U_N$$

where

$$\mathcal{S} = \mathcal{X} \times_1 U_1^T \times_2 U_2^T \ldots \times_N U_N^T.$$

Moreover,

$$\|S_{(n)}(i,:)\|_2 = \sigma_i(X_{(n)}) \quad i = 1, \ldots, \mathit{rank}(X_{(n)}).$$

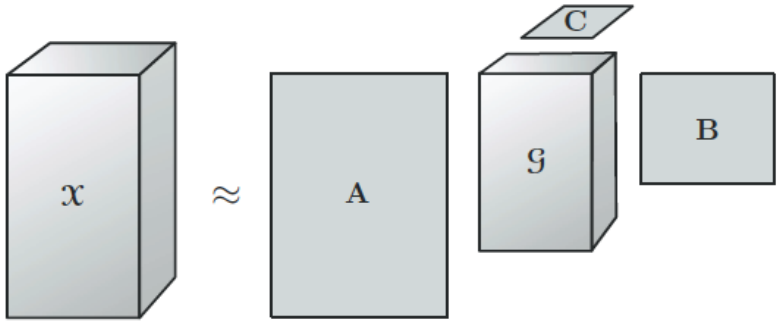

FIGURE 4.2. Tucker decomposition of a third order tensor

Let $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$. Then n**-rank** of $\mathcal{X}$ is the column rank of $X_{(n)}$. We denote the n-rank of $\mathcal{X}$ by $R_n = \text{rank}_n(\mathcal{X})$ for $n = 1, \ldots, N$ and we say that $\mathcal{X}$ is a rank-$(R_1, R_2, \ldots, R_N)$ tensor. The idea of n-rank was first introduced by Kruskal [42] and was developed by De Lathauwer [20] later. For a given tensor $\mathcal{X}$, the Tucker decomposition of rank $(R_1, R_2, \ldots, R_N)$, where $R_n = \text{rank}_n(\mathcal{X})$ can be found easily. The Figure (4.2) illustrates the Tucker decomposition of a third order tensor. In general, Tucker decomposition is not unique. Consider a third order tensor $\mathcal{X}$ with factor matrices A, B, C and the core tensor $\mathcal{S}$, then for any invertiable matrices $U \in \mathbb{R}^{P \times P}, V \in \mathbb{R}^{Q \times Q}$ and $W \in \mathbb{R}^{R \times R}$ we have

$$\mathcal{X} \approx [\mathcal{S}; A, B, C] = [\mathcal{S} \times_1 U \times_2 V \times_3 W; AU^{-1}, BV^{-1}, CW^{-1}].$$

So we can modify the core matrix $\mathcal{S}$ without worsening the fit by correct modifications to the factor matrices. One way to resolve this non uniqueness property of Tucker decomposition is to impose an extra condition on the core tensor $\mathcal{S}$. For instance we can obtain the unique core with the most zero elements [74].

4.3. Alternating Least Squares (ALS)

Recall that the CP decomposition of a tensor decomposes a given tensor into the sum of a rank one tensors, i.e.

$$\mathcal{X} \approx \sum_{r=1}^{R} a_r^{(1)} \circ \ldots \circ a_r^{(N)}. \tag{4.10}$$

In order to measure the approximation term in (4.2) the Frobenius norm of a tensor is used, so the problem is to minimize the Frobenius norm difference of $\mathcal{X}$ and the summation of the rank one tensors. In particular, the related optimization problem has the following form

$$\min_{a_r^{(1)},\ldots,a_r^{(N)}} \frac{1}{2} \left\| \mathcal{X} - \sum_{r=1}^{R} a_r^{(1)} \circ \ldots \circ a_r^{(N)} \right\|_F^2 \tag{4.11}$$

applying the factor matrices notation the optimization problem has the following form

$$\min_{A^{(1)},\ldots,A^{(N)}} \frac{1}{2} \|\mathcal{X} - [A^{(1)}, \ldots, A^{(N)}]\|_F^2 \tag{4.12}$$

In order to introduce the ALS algorithm, we need the following lemma first. The lemma states the relation between the modal unfolding (matricization) of the summation of rank one tensors and the Khatri-Rao product of the factor matrices.

LEMMA 4.1. *Let*

$$\mathcal{X} = \sum_{r=1}^{R} a_r^{(1)} \circ \ldots \circ a_r^{(N)},$$

and

$$A^{(n)} = \begin{pmatrix} a_1^{(n)} & \ldots & a_R^{(n)} \end{pmatrix}, \qquad n = 1, \ldots, N.$$

Then we have

$$X_{(n)} = A^{(n)} \left(A^{(N)} \odot \ldots \odot A^{(n+1)} \odot A^{(n-1)} \odot \ldots \odot A^{(1)} \right)^T \tag{4.13}$$

for all $n = 1, \ldots, N$.

PROOF. Note that for $n \in \{a, \ldots, N\}$ we have

$$\begin{aligned} X_{(n)} &= \left(\sum_{r=1}^{R} a_r^{(1)} \circ \ldots \circ a_r^{(N)} \right)_{(n)} \\ &= \sum_{r=1}^{R} \left(a_r^{(1)} \circ \ldots \circ a_r^{(N)} \right)_{(n)} \end{aligned}$$

using the equation (2.1) the last equality becomes

$$\begin{aligned} X_{(n)} &= \sum_{r=1}^{R} a_r^{(n)} \left(a_r^{(N)} \otimes \ldots \otimes a_r^{(n+1)} \otimes a_r^{(n-1)} \otimes \ldots \otimes a_r^{(1)} \right)^T \\ &= \begin{pmatrix} a_1^{(n)} & \ldots & a_R^{(n)} \end{pmatrix} \begin{pmatrix} a_1^{(N)} \otimes \ldots \otimes a_1^{(1)} & \ldots & a_R^{(N)} \otimes \ldots \otimes a_R^{(1)} \end{pmatrix}^T \\ &= A^{(n)} (A^{(N)} \odot \ldots \odot A^{(n+1)} \odot A^{(n-1)} \odot \ldots \odot A^{(1)})^T \end{aligned}$$

□

We also see that the Frobenius norm of a tensor is equal to the Frobenius norm of any of its modal unfoldings (matricizations), hence

$$\|\mathcal{X} - [A^{(1)}, \ldots, A^{(N)}]\|_F = \|X_{(n)} - [A^{(1)}, \ldots, A^{(N)}]_{(n)}\|_F, \tag{4.14}$$

for $n \in \{1, \ldots, N\}$. This relation comes useful in the process of the ALS algorithm which we describe in the next section.

4.3.1. The ALS Scheme. In this subsection we review the ALS scheme. ALS is an iterative method for finding the CP decomposition of a given tensor. It is effective, fast and easy to implement. For the purpose of simplicity, first we look at only the third order tensors, later the we describe the algorithm for arbitrary order N tensors. The problem we want to solve is: given a third order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ and positive integer R we want to calculate the CP decomposition of $\mathcal{X}$ such the residual is minimized, i.e.

$$\min \frac{1}{2} \|\mathcal{X} - \mathcal{Y}\|_F^2 \quad \text{s.t.} \quad \mathcal{Y} = \sum_{r=1}^{R} a_r \circ b_r \circ c_r.$$

This is equivalent to

$$\min_{a_r,b_r,c_r} \frac{1}{2}\|\mathcal{X} - \sum_{r=1}^{R} a_r \circ b_r \circ c_r\|_F^2 \tag{4.15}$$

By the earlier discussion in the introduction of this section, the equation above has three equivalent matricization forms. So the objective function to be minimized can be expressed in three following expressions:

$$\min_{A,B,C} \frac{1}{2}\|X_{(1)} - A(C \odot B)^T\|_F^2,$$

$$\min_{A,B,C} \frac{1}{2}\|X_{(2)} - B(C \odot A)^T\|_F^2,$$

and

$$\min_{A,B,C} \frac{1}{2}\|X_{(3)} - C(B \odot A)^T\|_F^2.$$

The ALS (Alternating Least Squares) solves the optimization problem (4.15) by fixing all factor matrices but one each time. Thus the problem reduces to a linear least squares problem at each iteration, starting with the initial guess A^0, B^0, C^0, the sequences A^k, B^k, C^k generated by ALS algorithm are,

$$A^{k+1} = \underset{A\in\mathbb{R}^{I\times R}}{\operatorname{argmin}} \frac{1}{2}\|X_{(1)} - A(C^k \odot B^k)^T\|_F^2,$$

$$B^{k+1} = \underset{B\in\mathbb{R}^{J\times R}}{\operatorname{argmin}} \frac{1}{2}\|X_{(2)} - B(C^k \odot A^{k+1})^T\|_F^2, \tag{4.16}$$

and

$$C^{k+1} = \underset{C\in\mathbb{R}^{K\times R}}{\operatorname{argmin}} \frac{1}{2}\|X_{(3)} - C(B^{k+1} \odot A^{k+1})^T\|_F^2.$$

Each of the equations in (4.16) is a linear least squares problem, thus the ALS algorithm (3) first fixes B and C and solves for A, next it fixes A and C and solves for B, finally it fixes A and B and solves for C. The notation / in MATLAB calculates the solution of the equation $AX = B$, i.e. B/A means BA^{-1} or $BA^{\dagger}$. The process of generating the sequence A^k, B^k, C^k continues until the provided stopping criteria is met.

Algorithm 3 The ALS Algorithm

Initialize with factor matrices A^0, B^0, C^0 where $A^0 \in \mathbb{R}^{I\times R}, B^0 \in \mathbb{R}^{J\times R}$ and $C^0 \in \mathbb{R}^{K\times R}$

General Step For $k = 1, 2, \ldots$ update A, B and C:

$$A^{k+1} = \underset{A\in\mathbb{R}^{I\times R}}{\operatorname{argmin}} \frac{1}{2}\|X_{(1)} - A(C^k \odot B^k)^T\|_F^2,$$
$$B^{k+1} = \underset{B\in\mathbb{R}^{J\times R}}{\operatorname{argmin}} \frac{1}{2}\|X_{(2)} - B(C^k \odot A^{k+1})^T\|_F^2,$$
$$C^{k+1} = \underset{C\in\mathbb{R}^{K\times R}}{\operatorname{argmin}} \frac{1}{2}\|X_{(3)} - C(B^{k+1} \odot A^{k+1})^T\|_F^2.$$

End for.

Now consider the general case $\mathcal{X} \in \mathbb{R}^{I_1\times\ldots\times I_N}$ with a given rank R. In each iteration the ALS algorithm fixes all the factor matrices but one and solve the sub-problem over it. Hence the optimization problem to solve is the following

$$\min_{\mathcal{Y}} \frac{1}{2}\|\mathcal{X} - \mathcal{Y}\|_F^2, \quad \text{s.t.} \quad \mathcal{Y} = \sum_{r=1}^{R} a_r^{(1)} \circ \ldots \circ a^{(N)}. \tag{4.17}$$

Similar to the case of third order tensor, the problem above has N equivalent forms which can be obtained by N different modal unfoldings of tensor $\mathcal{X}$, we have

$$Y_{(n)} = A^{(n)}(A^{(N)} \odot \ldots \odot A^{(n+1)} \odot A^{(n-1)} \odot \ldots \odot A^{(1)})^T.$$

Therefore in order to update the n-th factor matrix $A^{(n)}$, we can use the following

$$\left(A^{(n)}\right)^{k+1} = X_{(n)}/((A^{(N)})^k \odot \ldots \odot (A^{(n+1)})^k \odot (A^{(n-1)})^{k+1} \odot \ldots \odot (A^{(1)})^{k+1})^T,$$

and generate the factor matrices sequence $(A^{(n)})^k$. The algorithm is shown in (4).

4.4. ALS as an Optimization Problem

In this section, we study the CP decomposition problem (4.17) and the ALS algorithm in the framework of nonlinear optimization problem. Let f be the objective

Algorithm 4 The ALS Algorithm in General Case

Initialize with factor matrices A^0, B^0, C^0 where $A^0 \in \mathbb{R}^{I\times R}, B^0 \in \mathbb{R}^{J\times R}$ and $C^0 \in \mathbb{R}^{K\times R}$

General Step For $k = 1, 2, \ldots$:

For $n = 1, \ldots, N$ update the N factor matrices

$$\left(A^{(n)}\right)^{k+1} = \underset{A^{(n)}\in\mathbb{R}^{I_n\times R}}{\operatorname{argmin}} \frac{1}{2}\|X_{(n)} - A((A^{(N)})^k \odot \ldots \odot (A^{(n+1)})^k \circ (A^{(n-1)})^{k+1} \odot \ldots \odot (A^{(1)})^{k+1})^T\|_F^2,$$

End for

End for.

function in (4.17), i.e.

$$f : \mathbb{R}^{I_1\times R} \otimes \ldots \otimes \mathbb{R}^{I_N\times R} \to \mathbb{R}^+,$$

$$f(A^{(1)}, \ldots, A^{(N)}) = \frac{1}{2}\|\mathcal{X} - [A^{(1)}, \ldots, A^{(N)}]\|_F^2. \tag{4.18}$$

Note that the function f in (4.18) is continuous and differentiable, but is not convex. It can be written in N different equivalent forms using N different modal unfoldings. We can think of ALS as a block non-linear Gauss-Seidel method for solving (4.18). Hence at each iteration, the goal is to solve

$$\min_{A^{(n)}} f(A^{(1)}, \ldots, A^{(N)}), \tag{4.19}$$

for a fixed n, whie holding all other factor matrices constant. By matricization of $\mathcal{X}$, we can rewrite the equation above in matrix form

$$\min_{A^{(n)}} \frac{1}{2}\|X - A^{(n)}(A^{(-n)})^T\|_F^2. \tag{4.20}$$

Since only one factor matrix is the variable, the sub problems are linear least squares and they have the exact solutions:

$$A^{(n)} = X_{(n)}\left((A^{(-n)})^T\right)^\dagger. \tag{4.21}$$

However, this requires calculating the pseudo-inverse of a matrix which has the size $\prod_{m\neq n}^{N} I_m \times R$ which could be computationally expensive. The relations stated in (2.3) will come useful here. Define

$$\gamma^{(n)} = (A^{(n)})^T A^{(n)}, \quad \text{for } n = 1, \ldots, N.$$

then we have

$$A^{(n)} = X_{(n)} A^{(-n)} (\Gamma^{(n)})^{\dagger},$$

where

$$\Gamma^{(n)} = \gamma^{(1)} * \ldots * \gamma^{(n-1)} * \gamma^{(n+1)} * \ldots \gamma^{(N)}.$$

As we can see the new formulation only requires calculation the pseudo-inverse of a matrix of size $R \times R$ at each inner iteration in the ALS procedure. Next, we calculate the gradient of the function f defined in (4.18) with respect to the column vectors $a_r^{(n)}$. Note that $a_r^{(n)}$ is a vector of size I_n, so the partial derivative $\frac{\partial f}{\partial a_r^{(n)}}$ is also a vector of size I_n.

LEMMA 4.2. *[1] The partial derivatives of the objective function f defined in (4.18) are*

$$\frac{\partial f}{\partial a_r^{(n)}} = -\left(\mathcal{X} \times_{m\neq n}^{N} a_r^{(m)}\right) + \sum_{l=1}^{R} \gamma_{rl}^{(n)} a_l^{(n)}, \tag{4.22}$$

for $r = 1, \ldots, R$ and $n = 1, \ldots, N$, with $\gamma_{rl}^{(n)}$ defined as

$$\gamma_{lr}^{(n)} = \prod_{m\neq n}^{N} (a_r^{(m)})^T a_l^{(m)}.$$

A direct consequence of the lemma (4.2) provides the partial derivatives of f with respect to the factor matrices $A^{(n)}$. Note that the factor matrix $A^{(n)}$ is of size $I_n \times R$, therefore the partial derivative $\frac{\partial f}{\partial A^{(n)}}$ is also of size $I_n \times R$.

COROLLARY 4.1. *The partial derivatives of the objective function f in (4.18) are given by*

$$\frac{\partial f}{\partial A^{(n)}} = -X_{(n)} A^{(-n)} + A^{(n)} \Gamma^{(n)}, \tag{4.23}$$

for $n = 1, \ldots, N$.

PROOF. We can rewrite the equation (4.22) as

$$\frac{\partial f}{\partial a_r^{(n)}} = -X_{(n)} a^{(-n)} + A^{(n)} \gamma_r^{(n)},$$

for $r = 1, \ldots, R$. But this represents the r-th column of $\frac{\partial f}{\partial A^{(n)}}$, therefore we must have

$$\frac{\partial f}{\partial A^{(n)}} = -X_{(n)} A^{(-n)} + A^{(n)} \Gamma^{(n)}.$$

□

COROLLARY 4.2. *The partial derivatives of the objective function* f *in (4.18) are given by*

$$\frac{\partial f}{\partial A^{(n)}} = -X_{(n)} A^{(-n)} + A^{(n)} (A^{(-n)})^T A^{(-n)}, \tag{4.24}$$

for $n = 1, \ldots, N$.

PROOF. The result follows from (4.1) and the following relation between the Khatri-Rao product and Hadamard product of matrices:

$$(A \odot B)^T (A \odot B) = A^T A * B^T B.$$

□

The ALS algorithm can be described in the view of alternating block minimization technique which minimizes a function over a certain block at each inner iteration. In each inner iteration we solve the following minimization problem

$$\min_{A^{(n)}} f(A^{(1)}, \ldots, A^{(N)}), \tag{4.25}$$

and the minimum point is attained at a stationary point. Hence we solve the sub problem (4.25) by setting the gradient $\nabla_{A^{(n)}} f$ equal to zero. This implies the following normal equation

$$\nabla_{A^{(n)}} f = 0 \quad \Rightarrow \quad X_{(n)} A^{(-n)} = A^{(n)} \Gamma^{(n)},$$

or equivalently

$$X_{(n)}A^{(-n)} = A^{(n)}(A^{(-n)})^T A^{(-n)}.$$

The Hessian matrix of f can be calculated also [1] but we will not use the second order methods in any of our computations and we will not state it here.

4.5. CP Decomposition as a Nonlinear Least Squares Problem

The formulation of CP as a nonlinear least squares was first introduced by Paatero [55]. Consider the CP objective function f in (4.18). It can be rewritten as

$$f : \mathbb{R}^P \to \mathbb{R},$$

where

$$P = R\sum_{n=1}^{N} I_n,$$

and the argument of f is obtained by rearranging the factor matrices as follow

$$x = \begin{pmatrix} a_1^{(1)} \\ \vdots \\ a_R^{(N)} \end{pmatrix}.$$

For simplicity, consider the case of third order tensors where $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$. We can define the residual function $F : \mathbb{R}^P \to \mathbb{R}^Q$, where

$$Q = I \times J \times K,$$

and

$$F_{\alpha(i_1,i_2,i_3)} = x_{i_1 i_2 i_3} - \sum_{r=1}^{R} a_r(i_1)b_r(i_2)c_r(i_3),$$

for $\alpha(i_1, i_2, i_3) = i_1 + (i_2 - 1)I + (i_3 - 1)IJ$. By a straightforward calculation the Jacobian matrix of F can be expressed in a highly structured block form:

$$J = \begin{pmatrix} J_a & J_b & J_c \end{pmatrix},$$

$$J_a = \begin{pmatrix} J_a^1 & J_a^2 & \dots & J_a^R \end{pmatrix},$$

$$J_b = \begin{pmatrix} J_b^1 & J_b^2 & \dots & J_b^R \end{pmatrix},$$

$$J_c = \begin{pmatrix} J_c^1 & J_c^2 & \dots & J_c^R \end{pmatrix},$$

where

$$J_a^r = -c_r \otimes b_r \otimes I, \quad J_b^r = -c_r \otimes I \otimes a_r, \quad J_c^r = -I \otimes b_r \otimes a_r.$$

As an example let $\mathcal{X} \in \mathbb{R}^{2\times 2\times 2}$ and $R = 1$, i.e. $\mathcal{X} \approx a \circ b \circ \circ c$, then the Jacobian matrix $J \in \mathbb{R}^{8\times 6}$ looks like

$$J = -\begin{pmatrix} b_1c_1 & 0 & a_1c_1 & 0 & a_1b_1 & 0 \\ 0 & b_1c_1 & a_2c_1 & 0 & a_2b_1 & 0 \\ b_2c_1 & 0 & 0 & a_1c_1 & a_1b_2 & 0 \\ 0 & b_2c_1 & 0 & a_2c_1 & a_2b_2 & 0 \\ b_1c_2 & 0 & a_1c_2 & 0 & 0 & a_1b_1 \\ 0 & b_1c_2 & a_2c_2 & 0 & 0 & a_2b_1 \\ b_2c_2 & 0 & 0 & a_1c_2 & 0 & a_1b_2 \\ 0 & b_2c_2 & 0 & a_2c_2 & 0 & a_2b_2 \end{pmatrix}_{8\times 6}.$$

As we can see for the third order tensor $\mathcal{X}$, the Jacobian matrix is sparse with only three nonzero elements on each row. This is true in general, i.e. the matrix J will have only NR nonzero entries on each row. It is very tall and sparse. For instance, if $\mathcal{X}$ is a tensor of size $3 \times 4 \times 4$ with $R = 2$ components, then the Jacobian matrix has $Q = \prod_{n=1}^{3} I_n = 48$ rows, $P = R\sum_{n=1}^{3} I_n = 22$ columns, and $NRQ = 360$ nonzero entries. Figure (4.3) shows the sparsity structure of the Jacobian matrix.

Using the Jacobian, we can apply different methods to solve the CP problem in the nonlinear least squares sense. Since the Jacobian matrix is rank deficient, the Gauss-Newton method can cause problems while solving the related linear system. On the other hand, the Jacobian can be very large in size ($Q \times P$) and solving the related linear system can be computationally expensive. Thus it is preferable to work with J^TJ instead [72] and apply the Levenberg-Marquardt algorithm which

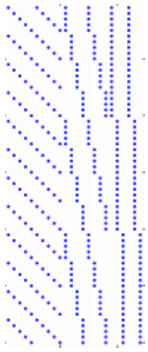

FIGURE 4.3. The sparsity of the Jacobian matrix. The blue points show the nonzero elements of J.

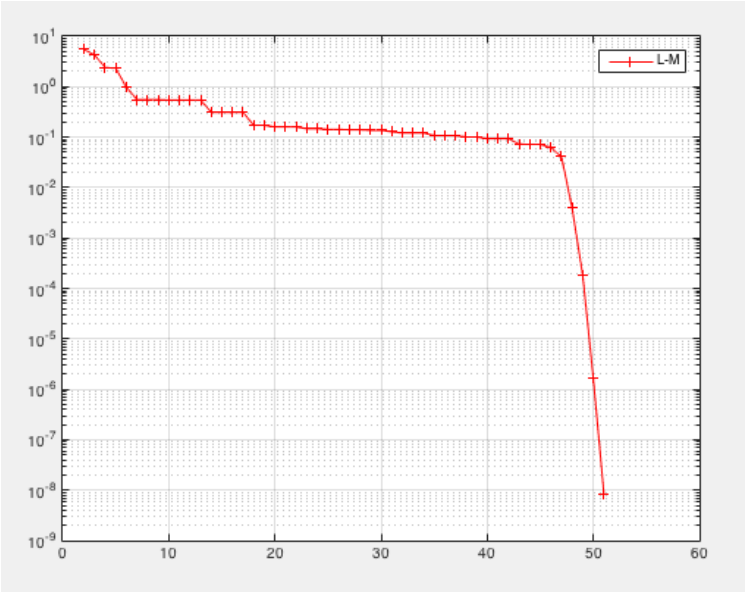

FIGURE 4.4. The residual error of the objective function in CP problem versus the number of iterations in Levenberg-Marquardt algorithm

was introduced in chapter 3. Figure (4.4) shows the performance of the Levenberg-Marquardt method to find the CP decomposition of a tensor of size $6 \times 7 \times 8$ with $R = 6$ rank one components.

The nonlinear least squares methods are faster than ALS in a sense of number of iterations, but they require to solve a very large linear system when the size of tensor becomes large. They also provide more accurate fit for a given tensor, furthermore

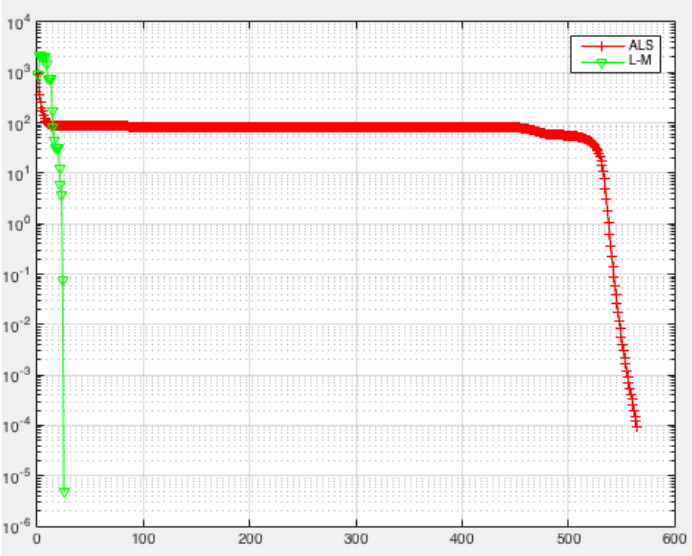

FIGURE 4.5. The comparison between the ALS algorithm and Levenberg-Marquardt on a randomly generated tensor of size $6 \times 7 \times 8$ with rank 6. The x axis represents the number of iteration and the y axis represents the error of the residual function in CP formulation.

in comparison to ALS they often do not encounter swamp. Figure (4.5) compares the performance of Levenberg-Marquardt method versus the ALS algorithm in the presence of swamp.

As we can see from (4.5) the alternating least squares algorithm takes a long time to converge. It requires more that 500 iterations to finally starts decreasing the residual error. The flat red line in the log error plot indicates the swamp phenomenon [49] . However the Levenberg-Marquardt (the green curve) requires only 20 iterations to converge.

CHAPTER 5

Rank Approximation of Tensors

5.1. Introduction

In 1927, Hitchcock [30, 31] proposed the idea of the polyadic form of a tensor, i.e., expressing a tensor, multilinear array, as the sum of a finite number of rank-one tensors. This decomposition is called the canonical polyadic (CP) decompositon; it is known as CANDECOMP or PARAFAC. It has been extensively applied to many problems in various engineering [61, 62, 2, 22] and science [64, 39]. Specifically, tensor methods have been applied in many multidimensional datasets in signal processing applications [17, 18, 19], color image processing [80, 33] and video processing [65, 11]. Most of these applications rely on decomposing a tensor data into its low rank form to be able to perform efficient computing and to reduce memory requirements. In computer vision, detection of moving objects in video processing relies on foreground and background separation, i.e. the separation of the moving objects called foreground from the static information called background, requires low rank representation of video tensor. In color image processing, the rgb channels in color image representation requires extensions of the matrix models of gray-scale images to low rank tensor methods. There are several numerical techniques [16, 23, 36, 49, 61] for approximating a low rank tensor into its CP decomposition, but they do not give an approximation of the minimum rank. In fact, most low rank tensor algorithms require an a priori tensor rank to find the tensor decomposition. Several theoretical results [43, 45] on tensor rank can help, but they are limited to low-multidimensional and low order tensors.

In this work, the focus is on finding an estimation of the tensor rank and its rank-one tensor decomposition (CP) of a given tensor. There are also algorithms [17, 13]

which give tensor rank, but they are specific to symmetric tensor decomposition over the complex field using algebraic geometry tools. Our proposed algorithm addresses two difficult problems for the CP decomposition: (a) one is that finding the rank of tensors is a NP-hard problem [32] and (b) the other is that tensors can be ill-posed [21] and failed to have their best low-rank approximations.

The problem of finding the rank of a tensor can be formulated as a constrained optimization problem.

$$\min_{\alpha} \|\alpha\|_0 \text{ s.t. } \mathcal{X} = \sum_{r=1}^{R} \alpha_r (a_r \circ b_r \circ c_r)$$

where $\|\alpha\|_0$ represents the total number of non-zero elements of α. The rank optimization problem is NP hard and so to make it more tractable, the following formulation [77] is used:

$$\min_{A,B,C,\alpha} \frac{1}{2} \|\mathcal{X} - \sum_{r=1}^{R} \alpha_r (a_r \circ b_r \circ c_r)\|_F^2 + \gamma \|\alpha\|_1$$

where $\gamma > 0$ is the regularization parameter and the objective function is a composition of smooth and non-smooth functions. Our formulation includes a Tikhonov type regularization:

$$\min \frac{1}{2} \|\mathcal{X} - \sum_{r=1}^{R} \alpha_r (a_r \circ b_r \circ c_r)\|_F^2 + \frac{\lambda}{2} (\|A\|_F^2 + \|B\|_F^2 + \|C\|_F^2) + \gamma \|\alpha\|_1.$$

The added Tikhonov regularization has the effect of forcing the factor matrices to have the equal norm. Moreover, this formulation and its numerical methods described later give an overall improvement in the accuracy and thus, memory requirements of the tensor model found in [77].

5.1.1. Organization. Our paper is organized as follows. In Section 2, we provide some notations and terminologies used throughout this paper. In Section 3, we formulate an l_1-regularization optimization to the low-rank approximation of tensors. In Section 4, we describe a numerical method to solve the l_1-regularization optimization by using a proximal alternating minimization technique for the rank and an alternating least-squares for the decomposition. In Section 5, we provide an analysis

of convergence of the numerical methods. The numerical experiments in Section 6 consist of simulated low rank tensor, color images and videos. Finally, our conclusion and future work are given in Section 7.

5.2. Preliminaries

We denote the scalars in $\mathbb{R}$ with lower-case letters $(a, b, \ldots)$ and the vectors with lower-case letters $(a, b, \ldots)$. The matrices are written as upper-case letters $(A, B, \ldots)$ and the symbols for tensors are calligraphic letters $(\mathcal{A}, \mathcal{B}, \ldots)$. The subscripts represent the following scalars: $(\mathcal{A})_{ijk} = a_{ijk}$, $(A)_{ij} = a_{ij}$, $(a)_i = a_i$ and the r-th column of a matrix A is a_r. The matrix sequence is denoted $\{A^k\}$. An Nth order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a multidimensional array with entries $(\mathcal{X})_{i_1 i_2 \cdots i_N} = x_{i_1 i_2 \cdots i_N}$ for $i_k \in \{1, \ldots, I_k\}$ where $k \in 1, \ldots, N$. In particular, a third order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ is a multidimensional array with entries x_{ijk} for $i \in \{1, \ldots, I\}$, $j \in \{1, \ldots, J\}$ and $k \in \{1, \ldots, K\}$.

Here we present some standard definitions and relations in tensor analysis. The Kronecker product of two vectors $a \in \mathbb{R}^I$ and $b \in \mathbb{R}^J$ is denoted by $a \otimes b \in \mathbb{R}^{IJ}$:

$$a \otimes b = \left(a_1 b^T \ldots a_I b^T\right)^T.$$

The Khatri-Rao (column-wise Kronecker) product (see[64]) of two matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{K \times J}$ is defined as

$$A \odot B = (a_1 \otimes b_1 \ldots a_J \otimes b_J).$$

The outer product of three vectors $a \in \mathbb{R}^I$, $b \in \mathbb{R}^J$, $c \in \mathbb{R}^K$ is a third order tensor $\mathcal{X} = a \circ b \circ c$ with the entries defined as follows:

$$x_{ijk} = a_i b_j c_k.$$

Definition 5.2.1. *Given a matrix $W \in \mathbb{R}^{I \times J}$, the function vec : $\mathbb{R}^{I \times J} \to \mathbb{R}^{I \cdot J}$ where $vec(W) = v$ is a vector of size $I \cdot J$ obtained from column-stacking the column*

vectors of W; i.e.

$$vec(W)_l = v(l) = w_{ij},$$

where $l = j + (k-1)J$.

The vectorization of a third order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ is the process of transforming the tensor into a column vector, the vec : $\mathbb{R}^{I\times J\times K} \to \mathbb{R}^{IJK}$ map is defined as

$$\text{vec}(\mathcal{X})_{\beta(i,j,k)} = x_{ijk},$$

where $\beta(i,j,k) = i + (j-1)I + (k-1)IJ$. Using the definitions above, we get

$$\text{vec}(a \circ b \circ c) = c \otimes b \otimes a.$$

Definition 5.2.2 (Mode-n matricization). *Matricization is the process of reordering the elements of an Nth order tensor into a matrix. The mode-n matricization of a tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ is denoted by $\mathcal{X}_{(n)}$ and arranges the mode-n columns to be the columns of the resulting matrix. The mode-n column, $x_{i_1\cdots i_{n-1}:i_{n+1}\cdots i_N}$, is a vector obtained by fixing every index with the exception of the nth index.*

If we use a map to express such matricization process for any Nth order tensor $\mathcal{T} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, that is, the tensor element $(i_1, i_2, \ldots, i_N)$ maps to matrix element (i_n, j), then there is a formula to calculate j:

$$j = 1 + \sum_{\substack{k=1\\k\neq n}}^{N} (i_k - 1)J_k \quad \text{with} \quad J_k = \prod_{\substack{m=1\\m\neq n}}^{k-1} I_m.$$

For example, the tensor unfolding or matricization of a third order tensor $\mathcal{X}$ is the process or rearranging the elements of $\mathcal{X}$ into a matrix. The mode-n ($n = 1, 2, 3$) matricization is denoted by $\mathcal{X}_{(n)}$ and the elements of it can be expressed by the following relations:

$$\mathcal{X}_{(1)}(i,l) = x_{ijk}, \text{ where } l = j + (k-1)J \text{ and } \mathcal{X}_{(1)} \in \mathbb{R}^{I\times JK},$$

$$\mathcal{X}_{(2)}(j,l) = x_{ijk}, \text{ where } l = i + (k-1)I \text{ and } \mathcal{X}_{(2)} \in \mathbb{R}^{J\times KI},$$

and

$$\mathcal{X}_{(3)}(k,l) = x_{ijk}, \text{ where } l = i + (j-1)I \text{ and } \mathcal{X}_{(3)} \in \mathbb{R}^{K \times IJ}.$$

5.2.1. CP decomposition and the Alternating Least-Squares Method. In 1927, Hitchcock [30][31] proposed the idea of the polyadic form of a tensor, i.e., expressing a tensor as the sum of a finite number of rank-one tensors. Today, this decomposition is called the canonical polyadic (CP); it is known as CANDECOMP or PARAFAC. It has been extensively applied to many problems in various engineering [61, 62, 2, 22] and science [64, 39]. The well-known iterative method for implementing the sum of rank one terms is the Alternating Least-Squares (ALS) technique. Independently, the ALS was introduced by Carrol and Chang [15] and Harshman [29] in 1970. Among those numerical algorithms, the ALS method is the most popular one since it is robust. However, the ALS has some drawbacks. For example, the convergence of ALS can be extremely slow.

The CP decomposition of a given third order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ factorizes it to a sum of rank one tensors.

$$\mathcal{X} \approx \sum_{r=1}^{R} \alpha_r (a_r \circ b_r \circ c_r) \tag{5.1}$$

For simplicity we use the notation $[A, B, C, \alpha]_R$ to represent the sum on the right hand side of the equation above, where $A \in \mathbb{R}^{I \times R}$, $B \in \mathbb{R}^{J \times R}$ and $C \in \mathbb{R}^{K \times R}$ are called factor matrices.

$$A = [a_1 \dots a_R], \quad B = [b_1 \dots b_R], \quad C = [c_1 \dots c_R]$$

The CP decomposition problem can be formulated as an optimization problem. Given R the goal is to find vectors a_r, b_r, c_r, such that the distance between the tensor $\mathcal{X}$ and the sum of the outer products of a_r, b_r, c_r is minimized. The Frobenius norm (sum of squares of the entries) is mainly used to measure the distance.

$$\min_{A,B,C,\alpha} \frac{1}{2}\|\mathcal{X} - [A, B, C, \alpha]_R\|_F^2. \tag{5.2}$$

Using the Khatri-Rao product, the objective function in (5.2) can be stated in the following four equivalent forms:

$$\frac{1}{2}\|\mathcal{X}_{(1)} - A\,\mathrm{diag}(\alpha)(C \odot B)^T\|_F^2, \tag{5.3}$$

$$\frac{1}{2}\|\mathcal{X}_{(2)} - B\,\mathrm{diag}(\alpha)(C \odot A)^T\|_F^2, \tag{5.4}$$

$$\frac{1}{2}\|\mathcal{X}_{(2)} - C\,\mathrm{diag}(\alpha)(B \odot A)^T\|_F^2, \tag{5.5}$$

and

$$\frac{1}{2}\|\mathrm{vec}(\mathcal{X}) - \mathrm{vec}([A, B, C, \alpha]_R)\|_2^2. \tag{5.6}$$

All the functions in (5.3), (5.4), (5.5) and (5.6) are linear least squares problems with respect to matrices A, B, C and vector α. To find approximations to A,B,C, and α, these four optimization problems (5.3)-(5.6) are implemented iteratively and the minimizers are updated between each optimization problems (via Gauss-Seidel sweep) with a stopping criteria. This technique is called the Alternating Least Squares (ALS) Method. The ALS method is popular since it is robust and easily implementable. However, the ALS has some drawbacks. For example, the convergence of ALS can be extremely slow. Another drawback is the requirement of a tensor rank R before a CP decomposition is approximated. The next sections deal with tensor rank approximation.

5.3. Rank Approximation of a Tensor

The problem of finding the rank of a tensor can be formulated as a constrained optimization problem.

$$\min_{\alpha} \|\alpha\|_0 \text{ s.t. } \mathcal{X} = [A, B, C, \alpha]_R,$$

where $\|\alpha\|_0$ represents the total number of non-zero elements of α. Since the problem is NP hard [32], we replace $\|\alpha\|_0$ by the ℓ_1 norm of α. The ℓ_1 norm is defined as the sum of absolute value of the elements of α. So the rank approximation problem can be written as

$$\min_{\alpha} \|\alpha\|_1 \quad \text{s.t. } \mathcal{X} = [A, B, C, \alpha]_R$$

In order to obtain a CP decomposition of the given tensor $\mathcal{X}$ as well as the rank approximation, we formulate the rank approximation problem as follow:

$$\min_{A,B,C,\alpha} \frac{1}{2}\|\mathcal{X} - [A, B, C, \alpha]\|_F^2 + \gamma\|\alpha\|_1. \tag{5.7}$$

where $\gamma > 0$ is the regularization parameter. The objective function of the problem (5.7) is non-convex and non-smooth. However, it is a composition of a smooth and non-smooth functions.

Moreover, it is known that CP decomposition of a tensor is unique up to scaling anf permutation of factor matrices. Note that

$$[A, B, C, \alpha]_R = [cA, c^{-1}B, C, \alpha]_R,$$

for a nonzero scalar $c \in \mathbb{R}$. In order to overcome the scaling indeterminacy, we add a Tikhonov type regularization term to our objective function [36]. Let f and g be the following:

$$f(A, B, C, \alpha) = \frac{1}{2}\|\mathcal{X} - [A, B, C, \alpha]\|_F^2, \tag{5.8}$$

and

$$g(\alpha) = \gamma\|\alpha\|_1, \tag{5.9}$$

which represent the fitting term and the ℓ_1 regularization term in (5.7), then the rank approximation problem can be formulated as

$$\min f(A, B, C, \alpha) + \frac{\lambda}{2}(\|A\|_F^2) + \|B\|_F^2 + \|C\|_F^2) + g(\alpha). \tag{5.10}$$

The added Tikhonov regularization has the effect of forcing the factor matrices to have the equal norm. i.e.

$$\|A\|_F = \|B\|_F = \|C\|_F.$$

[55], Now let Ψ represent the objective function in (5.10) collectively, then

$$\Psi : \mathbb{R}^{R(I+J+K+1)} \to \mathbb{R}^+,$$

where

$$\Psi(A, B, C, \alpha) = f(A, B, C, \alpha) + \frac{\lambda}{2}(\|A\|_F^2 + \|B\|_F^2 + \|C\|_F^2) + g(\alpha). \tag{5.11}$$

Let $\omega = (A, B, C, \alpha)$, when B, C, α are fix, we represent $f(\omega)$ by $f(A)$ and $\Psi(\omega)$ by $\Psi(A)$.

5.4. Approximation of Tensor Rank in CP Decomposition

In this section we propose a block coordinate descent type algorithm for solving the problem (5.10). We consider four blocks of variables with respect to A, B, C and α. In particular, at each inner iteration, we solve the following minimization problems

$$A^{k+1} = \underset{A}{\operatorname{argmin}}\{f(A, B^k, C^k, \alpha^k) + \frac{\lambda}{2}\|A\|_F^2\}, \tag{5.12}$$

$$B^{k+1} = \underset{B}{\operatorname{argmin}}\{f(A^{k+1}, B, C^k, \alpha^k) + \frac{\lambda}{2}\|B\|_F^2\}, \tag{5.13}$$

$$C^{k+1} = \underset{A}{\operatorname{argmin}}\{f(A^{k+1}, B^{k+1}, C, \alpha^k) + \frac{\lambda}{2}\|C\|_F^2\}, \tag{5.14}$$

and

$$\alpha^{k+1} = \underset{\alpha}{\operatorname{argmin}}\{L_f^{\beta^k}(A^{k+1}, B^{k+1}, C^{k+1}, \alpha) + g(\alpha)\}, \tag{5.15}$$

where $L_f^{\beta^k}(\alpha)$ represents the proximal linearization [10] of f with respect to α, namely

$$L_f^{\beta^k}(\alpha) = \langle \alpha - \alpha^k, \nabla f_\alpha(\alpha^k)\rangle + \frac{1}{2\beta^k}\|\alpha - \alpha^k\|^2.$$

Note that each of the minimization problems in (5.12)-(5.15) is strictly convex, therefore A, B, C, α are uniquely determined at each iteration. In fact, the subproblems in (5.12)-(5.14) are standard liner least squares problems with an additional Tikhonov regularization term. One can see by vectorization of the objective functions, for instance, the residual term in (5.3) can be written as follows

$$\frac{1}{2}\|\text{vec}(\mathcal{X}_{(1)}) - (((C \odot B)\text{diag}(\alpha)) \otimes I)\text{vec}(A)\|_2^2.$$

Since the objective functions in (5.12)-(5.14) are strictly convex, the first order optimality condition is sufficient for a point to be minimum. In other words, the exact solutions of (5.12)-(5.14) can be given be the following normal equations

$$A(E^k(E^k)^T + \lambda I) = \mathcal{X}_{(1)}(E^k)^T,$$

$$B(F^k(F^k)^T + \lambda I) = \mathcal{X}_{(2)}(F^k)^T,$$

and

$$C(G^k(G^k)^T + \lambda I) = \mathcal{X}_{(3)}(G^k)^T,$$

where $E^k = \text{diag}(\alpha^k)(C^k \odot B^k)^T$, $F^k = \text{diag}(\alpha^k)(C^k \odot A^{k+1})^T$ and $G^k = \text{diag}(\alpha^k)(B^{k+1} \odot A^{k+1})^T$.

To update α in (5.15), we discuss the proximal operator first.

DEFINITION 5.4.1. *(proximal operator) Let $g : \mathbb{R}^n \to \mathbb{R}$ be a lower semicontinuous convex function, then the proximal operator of g with parameter $\beta > 0$ is defined as follow*

$$prox_{\beta g}(y) = \underset{x}{\text{argmin}}\{g(x) + \frac{1}{2\beta}\|x - y\|_2^2\}. \tag{5.16}$$

Using the proximal operator notation, the equation (5.15) is equivalent to

$$\alpha^{k+1} = \text{prox}_{\beta^k g}(\alpha^k - \beta^k \nabla_\alpha f(\alpha^k)).$$

Algorithm 1 The BCD algorithm to approximate rank

Input: A third order tensor $\mathcal{X}$, an upper bound R of rank($\mathcal{X}$), and the regularization parameters $\lambda > 0$, $\gamma > 0$ and the fixed stepsize β;

Output: An approximated tensor $\mathcal{Y}$ with an estimated rank $\hat{R}$;

Given initial guess $\mathcal{X}^0 = [A^0, B^0, C^0, \alpha^0]_R$.
while stopping criterion not met **do**
  Update A :
  $E = \text{diag}(\alpha)(C \odot B)^T$
  $A = (\mathcal{X}_{(1)}E)/(EE^T + \lambda I)$
  Update B :
  $F = \text{diag}(\alpha)(C \odot A)^T$
  $B = (\mathcal{X}_{(2)}F)/(FF^T + \lambda I)$
  Update C :
  $G = \text{diag}(\alpha)(B \odot A)^T$
  $C = (\mathcal{X}_{(3)}G)/(GG^T + \lambda I)$
  Update α :
  for $r = 1$ to R **do**
    $M(:, r) = \text{vec}(a_r \circ b_r \circ c_r)$
  end for
  $y = \alpha - \beta(M^T(M\alpha - \text{vec}(\mathcal{X})))$
  for $r = 1$ to R **do**

$$\alpha(r) = \begin{cases} y(r) - \beta & y(r) > \beta \\ 0 & |y(r)| \le \beta \\ y(r) + \beta & y(r) < -\beta \end{cases}$$

  end for
end while
Create tensor $\mathcal{Y}$ with factor matrices A, B, C and coefficients α.
Count the number of non-zero elements of α and assign it to $\hat{R}$.
return The tensor $\mathcal{Y}$ with the estimated rank $\hat{R}$.

FIGURE 5.1. The BCD algorithm for approximating the rank of a third order tensor.

This is easy to verify because

$$\begin{aligned} \alpha^{k+1} &= \text{prox}_{\beta g}(\alpha^k - \beta \nabla f_\alpha(\alpha^k)) \\ &= \underset{\alpha}{\text{argmin}}\{\frac{1}{2\beta}\|\alpha - \alpha^k + \beta \nabla_\alpha f(\alpha^k)\|_2^2 + g(\alpha)\} \\ &= \underset{\alpha}{\text{argmin}}\{L_f^\beta + g(\alpha)\}. \end{aligned}$$

REMARK 5.4.1. *The proximal operator in (5.16) is well-defined because the function $g(\alpha)$ is continuous and convex. Using the vec operator, we have*

$$vec([A, B, C, \alpha]_R) = \sum_{r=1}^{R} \alpha_r vec(a_r \circ b_r \circ c_r) \tag{5.17}$$

$$= \sum_{r=1}^{R} \alpha_r (c_r \otimes b_r \otimes a_r) \tag{5.18}$$

$$= M\alpha \tag{5.19}$$

where $M \in \mathbb{R}^{IJK \times R}$ is the matrix with columns $c_r \otimes b_r \otimes a_r$. Therefore we can rewrite the objective function f as

$$\frac{1}{2}\|vec(\mathcal{X}) - M\alpha\|_2^2 \tag{5.20}$$

It is easy to calculate the gradient of (5.20) with respect to α:

$$\nabla_\alpha f(A, B, C, \alpha) = M^T(M\alpha - \text{vec}(\mathcal{X})) \tag{5.21}$$

This implies the Lipschitz continuity of the gradient of f with respect to α. The Lipschitz constant is $Q_\alpha = \|M^T M\|$ so we must have

$$\|\nabla_\alpha f(\alpha_1) - \nabla_\alpha f(\alpha_2)\| \leq Q_\alpha \|\alpha_1 - \alpha_2\|. \tag{5.22}$$

5.5. Analysis of Convergence

in this section, we study the global convergence of the proposed algorithm under mild assumptions. The Kurdyka-Lojasiewicz [38], [50] property plays a key role in our analysis. We begin this section by stating the descent lemma.

LEMMA 5.1. *(Descent Lemma) Let $h : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable function, and ∇h is Lipschitz continuous with constand L, then for any $x, y \in \mathbb{R}^n$ we have*

$$h(x) \leq h(y) + \langle x - y, \nabla h(y)\rangle + \frac{L}{2}\|x - y\|^2.$$

Next lemma provides the theoretical estimate for the decrease in the objective function after a single update α.

LEMMA 5.2. *Suppose that α^{k+1} is obtained by the equation (5.4) and $0 < \beta^k < 1/Q_\alpha^k$, where Q_α^k's are defined in (5.22), then there is a constant $N^k > 0$ such that*

$$\Psi(\alpha^k) - \Psi(\alpha^{k+1}) \geq N^k \|\alpha^{k+1} - \alpha^k\|^2. \tag{5.23}$$

PROOF. Recall that

$$L_f^{\beta^k}(\alpha) = \langle \alpha - \alpha^k, \nabla f_\alpha(\alpha^k) \rangle + \frac{1}{2\beta^k} \|\alpha - \alpha^k\|^2,$$

and α^{k+1} is obtained by the equation

$$\alpha^{k+1} = \operatorname*{argmin}_\alpha \{L_f^{\beta^k}(\alpha) + g(\alpha)\},$$

therefore we must have

$$L_f^{\beta^k}(\alpha^{k+1}) + g(\alpha^{k+1}) \leq g(\alpha^k). \tag{5.24}$$

Since $\nabla_\alpha f$ is Lipschitz continuous with constant Q_α^k, by the descent lemma we have

$$\begin{aligned} f(\alpha^{k+1}) \leq f(\alpha^k) &+ \langle \alpha^{k+1} - \alpha^k, \nabla_\alpha f(\alpha^k) \rangle \\ &+ \frac{Q_\alpha^k}{2} \|\alpha^{k+1} - \alpha^k\|^2, \end{aligned}$$

with (5.24), the above inequality implies

$$\begin{aligned} f(\alpha^{k+1}) + g(\alpha^{k+1}) \leq f(\alpha^k) + g(\alpha^k) \\ - \left(\frac{1 - \beta^k Q_\alpha^k}{2\beta^k} \right) \|\alpha^{k+1} - \alpha^k\|^2. \end{aligned}$$

Setting

$$N^k = \frac{1 - \beta Q_\alpha^k}{2\beta} > 0,$$

proves the lemma. □

REMARK 5.5.1. *Suppose that Q_α^k's are bounded from above by the constant Q_α in the previous lemma, then for fixed step-size β where*

$$0 < \beta < 1/Q_\alpha,$$

we have

$$\Psi(\alpha^k) - \Psi(\alpha^{k+1}) \geq \left(\frac{1-\beta Q_\alpha}{2\beta}\right)\|\alpha^{k+1} - \alpha^k\|^2, \tag{5.25}$$

for each $k = 1, 2, \ldots$.

DEFINITION 5.5.1. *[27] A differentiable function* $h : \mathbb{R}^n \to \mathbb{R}$ *is called strongly convex if there is a constant* $\mu > 0$ *such that*

$$h(x) - h(y) \geq \langle \nabla h(y), x - y\rangle + \frac{\mu}{2}\|x - y\|^2,$$

for any $x, y \in \mathbb{R}^n$.

LEMMA 5.3. *Suppose that* A^{k+1} *is obtained by equation (5.12), then we have*

$$\Psi(A^k) - \Psi(A^{k+1}) \geq \frac{\lambda}{2}\|A^k - A^{k+1}\|_F^2.$$

PROOF. Note that the objective functions in (5.12) is strongly convex with parameter λ and by the first-order optimality condition we must have

$$\nabla_A f(A^{k+1}) + \lambda A^{k+1} = 0$$

now the strong convexity of $f + \|.\|_F^2$ yields

$$\begin{aligned} f(A^k) + \frac{\lambda}{2}\|A^k\|_F^2 - f(A^{k+1}) - \frac{\lambda}{2}\|A^{k+1})\|_F^2 \\ \geq \frac{\lambda}{2}\|A^k - A^{k+1}\|^2 \end{aligned}$$

which implies

$$\Psi(A^k) - \Psi(A^{k+1}) \geq \frac{\lambda}{2}\|A^k - A^{k+1}\|^2.$$

This proves the lemma. □

REMARK 5.5.2. *Similar results hold for the blocks* B *and* C*, if they are updated by equations (5.13) and (5.14). In particular, we have that*

The next theorem guarantees that the value of Ψ decreases monotonically at each iteration. This shows that the sequence $\{\omega^k\}$ generated by scheme (5.12), (5.13), (5.14) and (5.15) is monotonically decreasing in value,

THEOREM 5.1. *(Sufficient decrease property) Let Ψ represent the objective function in (5.11) and $\omega^k = (A^k, B^k, C^k, \alpha^k)$, then we have*

$$\Psi(\omega^k) - \Psi(\omega^{k+1}) \geq \rho \|\omega^k - \omega^{k+1}\|^2, \tag{5.26}$$

for some positive constant ρ. In addition we have

$$\sum_{k=0}^{\infty} \|\omega^k - \omega^{k+1}\|^2 < \infty. \tag{5.27}$$

PROOF. By lemmas 5.2 and 5.3 we have

$$\begin{aligned}\Psi(\omega^k) - \Psi(\omega^{k+1}) \geq \frac{\lambda}{2}(&\|A^k - A^{k+1}\|^2 + \|B^k - B^{k+1}\|^2 \\ &+ \|C^k - C^{k+1}\|^2) + N_\alpha \|\alpha^k - \alpha^{k+1}\|^2,\end{aligned}$$

setting $\rho = \min\{\lambda/2, N_\alpha\}$ gives the first result. This shows that the sequence $\{\Psi(\omega^k)\}$ generated by our algorithm is decreasing. The monotonicity of $\{\Psi(\omega^k)\}$ with the fact that Ψ is bounded from below, implies $\Psi(\omega^k) \to \inf \Psi = \underline{\Psi}$ as $k \to \infty$, next let $n > 2$ be a positive integer, then

$$\begin{aligned}\sum_{k=0}^{n-1} \|\omega^k - \omega^{k+1}\|^2 &\leq \frac{1}{\rho} \sum_{k=0}^{n-1} \left(\Psi(\omega^k) - \Psi(\omega^{k+1})\right) \\ &= \frac{1}{\rho}(\Psi(\omega^0) - \Psi(\omega^n)).\end{aligned}$$

Letting $n \to \infty$ proves the last statement. □

REMARK 5.5.3. *The sequence $\{\omega^k\}$ generated by the scheme (5.13)-(5.15) is bounded. The reason comes from the fact that unboundedness of $\{\omega^k\}$ occurs when at least one of the blocks A, B, C or α gets unbounded. This never happens due to the regularization terms in the objective function Ψ and the fact that $\Psi(\omega^k)$ is non-increasing.*

THEOREM 5.2. *Let $\{\omega^k\}_{k\in\mathbb{N}}$ be the sequence generated by our algorithm, then there exists a positive constant $\nu > 0$ such that for any $k \in \mathbb{N}$ there is a vector $\eta^{k+1} \in \partial\Psi(\omega^{k+1})$ such that*

$$\|\eta^{k+1}\| \leq \nu \|\omega^k - \omega^{k+1}\|.$$

PROOF. Let k be a positive integer. By equations (5.12), (5.13), (5.14) and the first order optimality condition we have

$$\nabla_A f(A^{k+1}, B^k, C^k, \alpha^k) + \lambda A^{k+1} = 0,$$

$$\nabla_B f(A^{k+1}, B^{k+1}, C^k, \alpha^k) + \lambda B^{k+1} = 0,$$

and

$$\nabla_C f(A^{k+1}, B^{k+1}, C^{k+1}, \alpha^k) + \lambda C^{k+1} = 0.$$

If we define

$$\eta_1^{k+1} = \nabla_A f(\omega^{k+1}) - \nabla_A f(A^{k+1}, B^k, C^k, \alpha^k),$$

then $\eta_1^{k+1} = \nabla_A \Psi(\omega^{k+1})$. similarly we can define vectors η_2^{k+1}, η_3^{k+1}. Next, by equation (5.15), we have that

$$\alpha^{k+1} = \operatorname*{argmin}_{\alpha} \{L_f^{\beta^k}(A^{k+1}, B^{k+1}, C^{k+1}, \alpha) + g(\alpha)\}. \tag{5.28}$$

Hence by the optimality condition, there exists $u^{k+1} \in \partial g(\alpha^{k+1})$ such that

$$\nabla_\alpha f(\alpha^k) + \frac{1}{\beta^k}(\alpha^{k+1} - \alpha^k) + u^{k+1} = 0.$$

Next, define

$$\begin{aligned} \eta_4^{k+1} &= \nabla_\alpha f(\omega^{k+1}) - \nabla_\alpha f(\alpha^k) + \frac{1}{\beta^k}(\alpha^k - \alpha^{k+1}) \\ &= \nabla_\alpha f(\omega^{k+1}) + u^{k+1}. \end{aligned}$$

Therefore $\eta_4^{k+1} \in \partial \Psi_\alpha(\omega^{k+1})$. From these facts we have that

$$\eta^{k+1} = (\eta_1^{k+1}, \eta_2^{k+1}, \eta_3^{k+1}, \eta_4^{k+1}) \in \partial \Psi(\omega^{k+1}).$$

We now estimate the norm of η^{k+1}. First note that by 5.5.3, $\{\omega^k\}$ is bounded and the objective function (without the ℓ_1 regularization term) is twice continuously differentiable, therefore as a consequence of mean value theorem, ∇f must be Lipschitz

continuous. Hence there must exist a constant P_1 such that

$$\begin{aligned}\|\eta_1^{k+1}\| &= \|\nabla_A f(\omega^{k+1}) - \nabla_A f(A^{k+1}, B^k, C^k, \alpha^k)\| \\ &\leq P_1\|\omega^k - \omega^{k+1}\|,\end{aligned}$$

similarly, constants P_2 and P_3 exist such that

$$\|\eta_2^{k+1}\| \leq P_2\|\omega^k - \omega^{k+1}\|,$$

and

$$\|\eta_3^{k+1}\| \leq P_3\|\omega^k - \omega^{k+1}\|.$$

Furthermore

$$\begin{aligned}\|\eta_f^{k+1}\|_2 &= \|\nabla_\alpha f(\omega^{k+1}) + u^{k+1}\| \\ &\leq \|\nabla_\alpha f(A^{k+1}, B^{k+1}, C^{k+1}, \alpha^{k+1}) - \nabla_\alpha f(A^{k+1}, B^{k+1}, C^{k+1}, \alpha^k)\| + \frac{1}{\beta^{k+1}}\|\alpha^{k+1} - \alpha^k\| \\ &\leq P_4\|\omega^{k+1} - \omega^k\|.\end{aligned}$$

Setting $\nu = \max\{P_1, P_2, P_3, P_4\}$, gives us the result. □

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function. The function f is said to have Kurdyka-Lojasiewicz (KL) property at point $\hat{x} \in \partial f$ if there exists $\theta \in [0, 1)$ such that

$$\frac{|f(x) - f(\hat{x})|^\theta}{\text{dist}(0, \partial f(x))}$$

is bounded around $\hat{x}$ [79]. A very rich class of functions satisfying the KL property is the semi-algebraic functions. These are functions where their graphs can be expressed as an algebraic set, that is

$$\text{Graph}(f) = \bigcup_{i=1}^{p}\bigcap_{j=1}^{q}\{x \in \mathbb{R}^n : P_{ij} = 0, Q_{ij}(x) > 0\},$$

where P_{ij}'s and Q_{ij}'s are polynomial functions and the graph of f is defined by

$$\text{Graph}(f) = \{(x, y) \in \mathbb{R}^n \times \mathbb{R} : f(x) = y\}.$$

Note that the univariate function $g(x) = |x|$ is semialgebraic because

$$\text{Graph}(g) = \{(x, y) : |x| = y\} = \{(x, y) : y - x = 0, x > 0\}$$
$$\cup \{(x, y) : y + x = 0, -x > 0\}.$$

The class of semi algebraic functions are closed under addition and composition [3]. Hence The objective function in (5.11) is semialgebraic therefore it satisfies KL property.

THEOREM 5.3. *Suppose that $\{\omega^k\}_{k\in\mathbb{N}}$ is the sequence generated by our algorithm, then $\{\omega^k\}_{k\in\mathbb{N}}$ converges to the critical point of Ψ.*

PROOF. By theorems (5.1) and (5.2), this is a direct result from theorem 2.9 in [3]. □

5.6. Numerical Experiment and Results

In this section we test our algorithm on tensors with different rank and dimensions. We randomly generate tensors with specified ranks and compare the performance of our algorithm with other available algorithms such as LRAT [77]. Next, we apply our algorithm on single moving object videos in order to extract the background and target object.

5.6.1. Tensor Rank Approximation. In this subsection we test the performance of our algorithm on randomly generated cubic tensors with various dimensions and various rank. The upper bound for the rank of tensors are set to be equal to $\min\{IJ, JK, IK\}$. The results are shown in TABLE I.

5.6.2. Comparison between LRAT and our algorithm. In this subsection, we compare the performance of our proposed algorithm to LRAT [77]. We generate a random cubic tensor $\mathcal{A} \in \mathbb{R}^{5\times5\times5}$ where its rank is equal to five. The comparison is based on the residual function as well as the sparsity of vector α. The upper bound for the rank of the tested tensor is set to be equal to ten for both algorithms. Figure

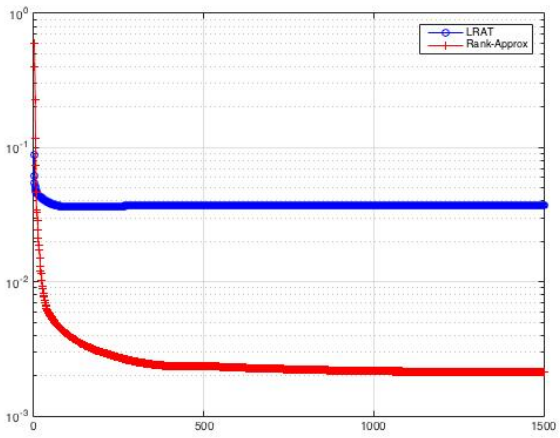

FIGURE 5.2. The comparison in the residual error of LRAT [77] against the proposed algorithm.

	Size of Tensor		
	$I, J, K = 5$	$I, J, K = 7$	$I, J, K = 10$
Actual Rank	5	8	10
Upper bound	10	15	20
Estimated Rank	5	8	12
Residual error	2.85e-1	1.34e-1	1.20e-1
Relative error	5.17e-2	1.05e-2	5.00e-3
Time	2.23	3.86	6.39

TABLE 5.1. Rank Approximation

(5.2) demonstrates the error of the fitting term $\|\mathcal{X} - \mathcal{X}_{est}\|_F$ during the first 1500 iterations for both algorithms. The x-axis represents the number of iterations and the y-axis represents the residual error. In terms of sparsity of vector α, the LRAT produces 4 zero components for α versus our algorithm which approximate the rank to be 5.

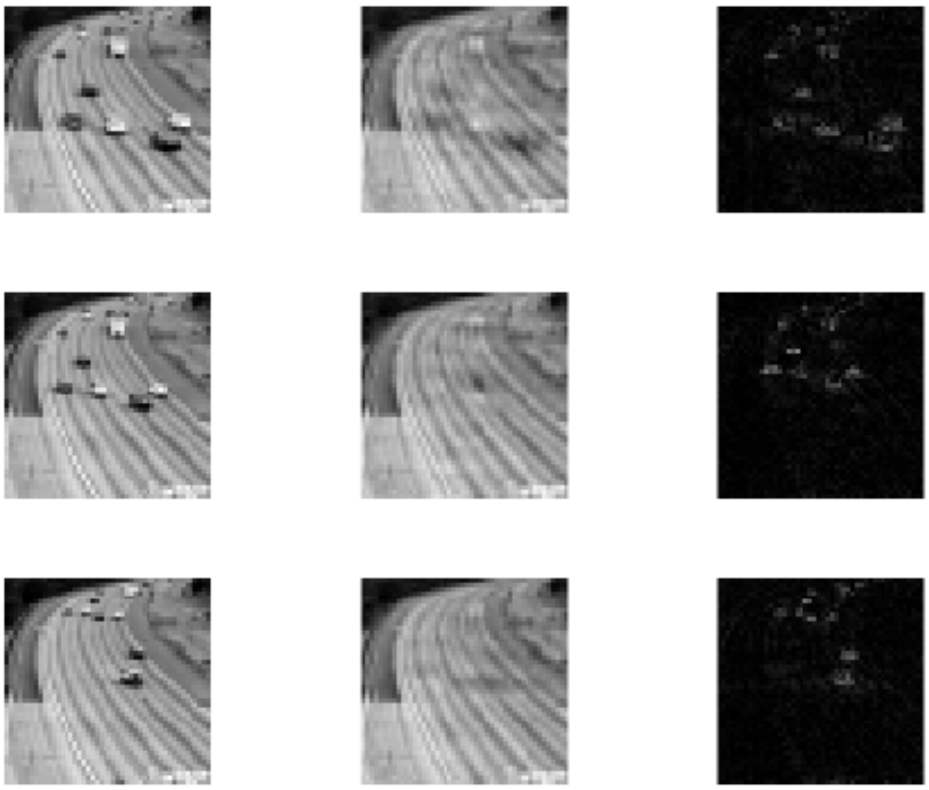

FIGURE 5.3. The original video [11, 65] is of the size $48 \times 48 \times 51$. Column 1 shows the original (11th,16th,49th) frames, column 2 shows the reconstruction (background) and column 3 shows the foreground (moving objects).

5.6.3. Application in background extraction of single moving object videos. In this subsection we apply our algorithm to extract the background of videos. See Figure 5.3. The video example [11, 65] is a $48 \times 48 \times 51$ with rank 23 tensor. The relative residual error of $\|\mathcal{X} - \sum_r^R \alpha_r a_r \circ b_r \circ c_r\|_F^2$ is 10^{-8}.

5.7. Conclusion

We presented the iterative algorithm for approximating tensor rank and CP decomposition based on a sparse optimization problem. Specifically, we apply a

(A) original frame

(B) background

(A) original frame

(B) background

(A) original frame

(B) background

FIGURE 5.4. The original video of this example is of size $240 \times 320 \times 500$. The left column represent the original sample frame taken from the original video and the right column represents the background extraction of the corresponding frame.

Tikhonov regularization method for finding the decomposition and a proximal algorithm for the tensor rank. We have also provided convergence analysis and numerical experiments on color images and videos. Overall, this new tensor sparse model and its computational method dramatically improve the accuracy and memory requirements.

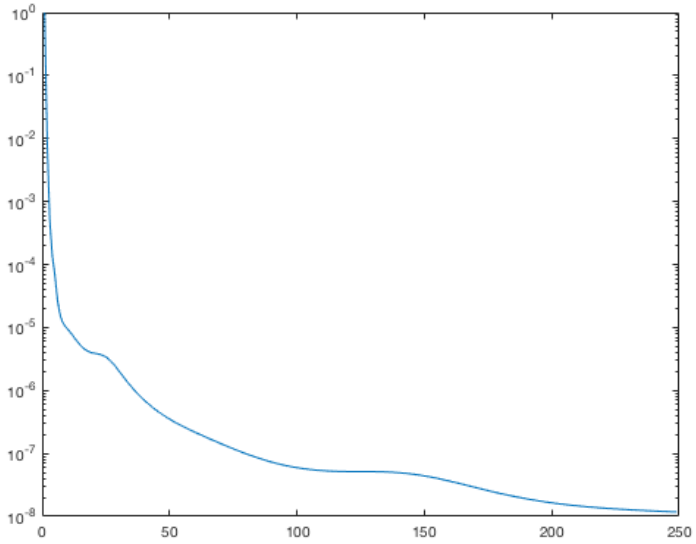

FIGURE 5.5. Residual Plot. The x-axis is the number of iterations and y-axis is the relative error term of $\|\mathcal{X} - \sum_r^R \alpha_r a_r \circ b_r \circ c_r\|_F^2$ for the video example in Figure 5.3.

(A) original image (B) reconstructed image

FIGURE 5.6. The performance of our algorithm on RGB image. The right image illustrates the compressed reconstructed version of the original image.

CHAPTER 6

Sampling Blocks in ALS

6.1. Introduction

Recall that the CP model can be formulated as an optimization problem. Let f be

$$f : \mathbb{R}^{I\times R} \otimes \mathbb{R}^{J\times R} \otimes \mathbb{R}^{K\times R} \to \mathbb{R}^{+}, \tag{6.1}$$

where

$$f(A, B, C) = \frac{1}{2}\|\mathcal{X} - [A, B, C]\|_F^2, \tag{6.2}$$

then the CP model minimizes f over the blocks A, B and C. In particular, the optimization problem becomes

$$\min_{A,B,C} f(A, B, C). \tag{6.3}$$

In general the problem in (6.3) is ill-posed [54], but there many ways to resolve this by adding additional constraints to the problem. For instance see [59] or [40]. The objective function in (6.3) is expressed in matrix form, However it can be written in vector form

$$f : \mathbb{R}^{P} \to \mathbb{R}^{+} \tag{6.4}$$

where $P = R(I + J + K)$. The argument of f is the vector x defined as follow

$$x = (a_1 \dots a_R, b_1 \dots b_R, c_1 \dots c_R)^T. \tag{6.5}$$

where a_r, b_r and c_r are the columns of the factor matrices A, B and C.

LEMMA 6.1. *The partial derivatives of the objetive function in (6.2) are given by*

$$\frac{\partial f}{\partial A} = -X_{(1)}(C \odot B) + A(C \odot B)^T(C \odot B), \tag{6.6}$$

$$\frac{\partial f}{\partial B} = -X_{(2)}(C \odot A) + B(C \odot A)^T(C \odot A) \tag{6.7}$$

and

$$\frac{\partial f}{\partial C} = -X_{(3)}(B \odot A) + C(B \odot A)^T(B \odot A). \tag{6.8}$$

PROOF. See Theorem 4.1 in [1]. □

REMARK 6.1.1. *The partial derivatives in (6.6), (6.7) and (6.8) are Lipschitz continuous.*

PROOF. We only show the Lipschitz continuity of (6.6), the other cases are quite similar. Note that for $A_1, A_2 \in \mathbb{R}^{I \times R}$, we have

$$\left\| \frac{\partial f}{\partial A}(A_1) - \frac{\partial f}{\partial A}(A_2) \right\| \leq \|A_1 - A_2\| \|(C \odot B)^T(C \odot B)\|. \tag{6.9}$$

This proves the Lipschitz continuity of the first partial derivative with Lipschitz constant $L_A = (C \odot B)^T(C \odot B)$. □

Recall that the ALS (Alternating Least Squares) algorithm solves the problem in (6.3) by fixing two blocks and minimizing over the third block alternatively, namely

$$A^{k+1} \in \operatorname*{argmin}_A \frac{1}{2}\|X_{(1)} - A(C^k \odot B^k)^T\|_F^2, \tag{6.10}$$

$$B^{k+1} \in \operatorname*{argmin}_B \frac{1}{2}\|X_{(2)} - B(C^k \odot A^{k+1})^T\|_F^2, \tag{6.11}$$

and

$$C^{k+1} \in \operatorname*{argmin}_C \frac{1}{2}\|X_{(3)} - C(B^{k+1} \odot A^{k+1})^T\|_F^2. \tag{6.12}$$

Hence, given the positive integer R and the initial guesses A^0, B^0, C^0, ALS generates the sequence A^k, B^k, C^k in order to recover the canonical decomposition of the given tensor.

6.2. Sampling the Rank-One Components in ALS

The ALS algorithm updates the factor matrices A, B, C at each inner iteration which means updating all rank-one components $a_r \circ b_r \circ c_r$ for $r = 1, \ldots, R$. However, this requires the computation of the pseudo inverse of an $R \times R$ matrix at each inner iteration. In order to reduce the computaion time we propose a method of sampling the rank-one components. In each iteration a subset S from $\{1, \ldots, R\}$ is chosen and the update of factor matrices is in accordance with S. Let $S \subset \{1, \ldots R\}$ be the set of sample indices and A_s, B_s and C_s represent the sub-matrices obtained by choosing the columns of A, B and C according to the index set S respectively. The partial derivatives of f with respect to the blocks A_S, B_S and C_S are

$$\frac{\partial f}{\partial A_S} = -X_{(1)}(C_S \odot B_S) + A(C \odot B)^T(C_S \odot B_S), \tag{6.13}$$

$$\frac{\partial f}{\partial B_S} = -X_{(2)}(C_S \odot A_S) + B(C \odot A)^T(C_S \odot A_S), \tag{6.14}$$

and

$$\frac{\partial f}{\partial C_S} = -X_{(3)}(B_S \odot A_S) + C(B \odot A)^T(B_S \odot A_S). \tag{6.15}$$

Setting $\nabla_{A_S} f$ equal to zero and solving for A_S implies

$$\begin{aligned} A_S \left((C \odot B)^T(C_S \odot B_S)\right)^S = -A_{S^C} \left((C \odot B)^T(C_S \odot B_S)\right)^{S^C} \\ + X_{(1)}(C_S \odot B_S), \end{aligned} \tag{6.16}$$

where A^S represents the sub matrix of A obtained by sampling the rows of A corresponding to the sampling set S. Similar results can be derived by solving the the equations $\nabla_{B_S} f = 0$ and $\nabla_{C_S} f = 0$. Hence, we have that

$$\begin{aligned} B_S \left((C \odot A)^T(C_S \odot A_S)\right)^S = -B_{S^C} \left((C \odot A)^T(C_S \odot A_S)\right)^{S^C} \\ + X_{(2)}(C_S \odot A_S), \end{aligned} \tag{6.17}$$

and

$$C_S\left((B \odot A)^T (B_S \odot A_S)\right)^S = -C_{S^C}\left((B \odot A)^T (B_S \odot A_S)\right)^{S^C} \tag{6.18}$$

$$+ X_{(3)}(B_S \odot A_S). \tag{6.19}$$

The stated normal equations are linear least squares and have exact solutions. Next, we show that how much decrease in the objective function is gained during each iteration. We start by stating the following lemma:

LEMMA 6.2. *(Descent Lemma) Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function whose gradient ∇f is Lipschitz continuous with constant L, then*

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}\|x - y\|^2 \tag{6.20}$$

for all $x, y \in \mathbb{R}^n$.

PROOF. We have

$$\begin{aligned}
f(x) - f(y) &= \int_0^1 \langle \nabla f(y + t(x - y)), x - y \rangle \, dt \\
&= \langle x - y, \nabla f(y) \rangle + \int_0^1 \langle \nabla f(y + t(x - y)) - \nabla f(y), x - y \rangle \, dt \\
&\leq \langle x - y, \nabla f(y) \rangle + \int_0^1 L\|t(x - y)\|\|x - y\| \, dt \\
&= \langle x - y, \nabla f(y) \rangle + L\|x - y\|^2 \int_0^1 t \, dt \\
&= \langle x - y, \nabla f(y) \rangle + \frac{L}{2}\|x - y\|^2.
\end{aligned}$$

□

In order to apply the descent lemma in our algorithm, we need to establish a block version of it. Assume that f is partitioned into p blocks, i.e.

$$x = (x(1), \ldots, x(p))^T,$$

Algorithm 5 Block ALS

Initialize $\omega^0 = (A^0, B^0, C^0)$ where $A^0 \in \mathbb{R}^{I\times R}, B^0 \in \mathbb{R}^{J\times R}$ and $C^0 \in \mathbb{R}^{K\times R}$

General Step select $S \subset \{1, \ldots, R\}$

For $k = 1, 2, \ldots$ updates the corresponding blocks A_S, B_S and C_S:

$$A_S^{k+1} \in \underset{A_S\in\mathbb{R}^{I\times|S|}}{\operatorname{argmin}} \frac{1}{2}\|X_{(1)} - (A_S, A_{SC}^k)(C^k \odot B^k)^T\|_F^2,$$

$$B_S^{k+1} \in \underset{B_S\in\mathbb{R}^{J\times|S|}}{\operatorname{argmin}} \frac{1}{2}\|X_{(2)} - (B_S, B_{SC}^k)(C^k \odot A^{k+1})^T\|_F^2,$$

$$C_S^{k+1} \in \underset{C_S\in\mathbb{R}^{K\times|S|}}{\operatorname{argmin}} C\frac{1}{2}\|X_{(3)} - (C_S, C_{SC}^k)(B^{k+1} \odot A^{k+1})^T\|_F^2.$$

Update A^{k+1}, B^{k+1} and C^{k+1} by replacing the columns A_S^{k+1}, B_S^{k+1} and C_S^{k+1}.

Set $\omega^{k+1} = (A^{k+1}, B^{k+1}, C^{k+1})$.

where $x(i) \in \mathbb{R}^{(n_i)}$ for $i = 1, \ldots, p$ and $n_1 + \ldots + n_p = n$. Define the matrices $U_i \in \mathbb{R}^{n\times n_i}$ where

$$(U_1, \ldots, U_p) = I_n,$$

note that $x(i) = U_i^T x$ for $x \in \mathbb{R}^n$ and $i = 1, \ldots, p$. By this setting we can rewrite vector $x \in \mathbb{R}^n$ in terms of its p blocks and matrices U_i:

$$x = \sum_{i=1}^{p} U_i x(i),$$

also the vector of partial derivatives corresponding to the block $x(i)$ can be expressed as

$$\nabla_i f(x) = U_i^T \nabla f(x), \quad i = 1, \ldots, p.$$

we say that the gradient of f is block i Lipschitz continuous with constant L_i if

$$\|\nabla_i f(x + U_i h_i) - \nabla_i f(x)\| \leq L_i \|h_i\|,$$

for every $h_i \in \mathbb{R}^{n_i}$. The constants L_i, $i = 1, \ldots, p$ are called the block Lipschitz constants. The following lemma is a direct consequence of the descent lemma.

LEMMA 6.3. *(Block Descent Lemma) Suppose that f is continuously differentiable with the block Lipschitz constants L_i. Assume that $u, v \in \mathbb{R}^n$ such that $v - u = U_i h$, for some $h \in \mathbb{R}^{n_i}$. Then*

$$f(v) \leq f(u) + \langle \nabla f(u), v - u \rangle + \frac{L_i}{2}\|u - v\|^2.$$

LEMMA 6.4. *Let A^k, B^k and C^k be the sequence of factor matrices which are obtained by Algorithm 1, then we must have*

$$f(A_S^k, :) - f(A_S^{k+1}, :) \geq \frac{1}{2L_S^k}\|\nabla f(A_S^k, :)\|_F^2. \tag{6.21}$$

Similar results hold for the block B and C.

PROOF. Note that

$$A_S^{k+1} \in \underset{A_S \in \mathbb{R}^{I \times |S|}}{\operatorname{argmin}} \frac{1}{2}\|X_{(1)} - A(C^k \odot B^k)^T\|_F^2, \tag{6.22}$$

therefore we must have

$$f(A_S^{k+1}, A_{S^C}^k, B^k, C^k) \leq f(A_S^k - \frac{1}{L_{A_S}^k}\nabla_{A_S} f(A_S^k), A_{S^C}^k, B^k, C^k) \tag{6.23}$$

which implies

$$f(A_S^k, :) - f(A_S^{k+1}, :) \geq f(A_S^k, :) - f(A_S^k - \frac{1}{L_{A_S}^k}\nabla_{A_S} f(A_S^k), :). \tag{6.24}$$

Hence by the block descent lemma we should have

$$f(A_S^k, :) - f(A_S^{k+1}, :) \geq \frac{1}{2L_{A_S}^k}\|\nabla_{A_S} f(A_S^k)\|_F^2 \tag{6.25}$$

Similarly, we have

$$f(B_S^k, :) - f(B_S^{k+1}, :) \geq \frac{1}{2L_{B_S}^k}\|\nabla_{B_S} f(B_S^k)\|_F^2, \tag{6.26}$$

and

$$f(C_S^k, :) - f(C_S^{k+1}, :) \geq \frac{1}{2L_{C_S}^k}\|\nabla_{C_S} f(C_S^k)\|_F^2. \tag{6.27}$$

□

The global convergence of algorithms depends on the sampling set S and the nature of ALS algorithm. The sampling method we adopt here, ensures that each coordinate block is chosen sufficiently often. The algorithm chooses each index at least once during R iterations. In other words, f $S_1, \ldots, S_R$ are the first choices of sampling, then we must have

$$S_1 \cup \ldots \cup S_R = \{1, \ldots, R\}.$$

Since ALS is a type of alternating minimization, the convergence is not guaranteed. The alternating minimization usually requires the strong convexity of the objective function. However, the global convergence of the proximal alternating minimization has been studied recently [4]. The proximal alternating minimization for ALS was proposed by Navasca et al. (2008). Later the global convergence of the algorithm was discussed in [78]. The analysis is based on the Kurdyka-ojasiewicz property of the objective function in CP model.

6.3. Numerical Experiments

In this section, we compare the performance of ALS against the sample block ALS. We give two examples of third-order tensor CP decomposition and one example on the RGB image compression. The codes of all three examples are written in MATLAB. In all numerical experiments the initial guesses are generated randomly (randn in MATLAB). All three experiments are run on a Mac with Intel i5 CPU 2.5 GHz and 4 GB memory.

EXAMPLE 6.3.1. *(Swamp in ALS) Let* $\mathcal{X} \in \mathbb{R}^{3\times3\times3}$ *be*

$$X(:,:,1) = \begin{pmatrix} -0.896 & -0.596 & 2.571 \\ -0.856 & -0.041 & 0.697 \\ -0.501 & 0.743 & -2.139 \end{pmatrix}, \; X(:,:,2) = \begin{pmatrix} 0.862 & 0.676 & -2.131 \\ -0.952 & 1.586 & -1.159 \\ -3.136 & 2.291 & 0.587 \end{pmatrix},$$

and

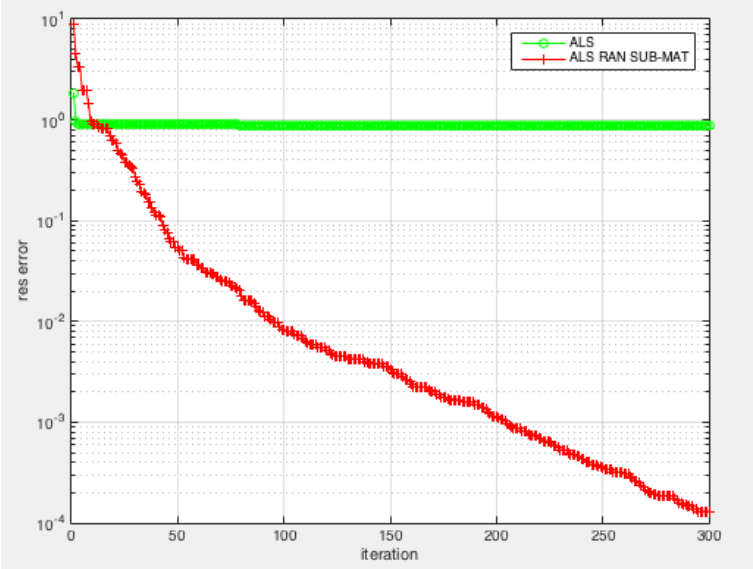

FIGURE 6.1. Plot of example (6.3.1).

$$X(:,:,3) = \begin{pmatrix} 1.366 & 0.514 & -5.816 \\ -0.405 & 0.967 & -2.806 \\ -2.719 & 1.258 & 2.343 \end{pmatrix}.$$

be the third order tensor in this example. The rank of $\mathcal{X}$ *is equal to three. We run both ALS and sample block ALS for this particular example with the same initial guesses for the factor matrices:*

$$A^0 = \begin{pmatrix} 0.059 & 0.806 & 0.930 \\ 0.231 & 0.038 & 0.045 \\ 0.935 & 0.968 & 0.329 \end{pmatrix}, \; B^0 = \begin{pmatrix} 0.716 & 0.306 & 0.115 \\ 0.031 & 0.834 & 0.139 \\ 0.242 & 0.521 & 0.404 \end{pmatrix},$$

and

$$C^0 = \begin{pmatrix} 0.159 & 0.227 & 0.720 \\ 0.944 & 0.026 & 0.204 \\ 0.129 & 0.890 & 0.340 \end{pmatrix}$$

Figure (6.1) demonstrates the comparison between the performance of ALS and the sample block ALS with only one rank-one component at each iteration. The plot shows the residual error $\|\mathcal{X} - [A^k, B^k, C^k]\|_F$ *versus the number of iterations. The*

Size of Tensor	*Rank*	*ALS Time*	*SBALS Time*	*ALS Res.*	*SBALS Res.*
$7 \times 7 \times 7$	*10*	*3.3e-3*	*6.3e-4*	*2.3e-1*	*4.9e-1*
$10 \times 10 \times 10$	*20*	*8.7e-3*	*1.4e-3*	*1.8*	*2.3*
$15 \times 15 \times 15$	*40*	*1.2e-2*	*6.7e-3*	*16.0*	*71.4*
$20 \times 20 \times 20$	*80*	*1.5e-1*	*2.3e-2*	*157.2*	*181.7*

TABLE 6.1. The comparison between ALS and sample block ALS in terms of execution time and residual function

maximum number of iterations for both methods are set to be 300. We can see from the plot that ALS does not improve the value of the residual error during the first 300 iterations. However, the sample block ALS has a constant decrease in the value of objective function during the iterations.

EXAMPLE 6.3.2. *Recall from chapter 3 that the ALS algorithm requires to compute the pseudo inverse of a $R \times R$ matrix at each inner iteration. In contrast, the sample block ALS computes the pseudo inverse of an $|S| \times |S|$ matrix at each inner iteration. In this example we compare the performance of ALS and the sample block ALS in terms of the required execution time in MATLAB for each algorithm. We generate cubic tensors of different sizes. The initial guesses for both methods are the same for all tensors. The maximum number iterations are set to be 1500 and the residual error tolerance is set to be 10^{-5}.*

Table (6.1) demonstrates the comparison between the ALS and the sample block ALS in terms of time and the fitting error. The time is measured in second for each inner iteration in both algorithms. The residual error is the $\|\mathcal{X} - [A, B, C]\|_F$. The size of sample $|S|$ is equal to $\lfloor R/4 \rfloor$. The ransom selections of S cycles through all the rank-one components of CP decomposition.

EXAMPLE 6.3.3. *In this example we apply the sample block ALS to an RGB image of size $200 \times 200 \times 3$. The approximated rank of the image is obtained by the proposed*

(A) ALS (B) SBALS

FIGURE 6.2. The left image is the compressed version of the original image by ALS. The right image is obtained by the SBALS.

FIGURE 6.3. The Original image for example 3.

algorithm in chapter 5. It is set to be $R_{est} = 62$ *for both ALS and sample block ALS. The size the sample is* $|S| = \lfloor R_{est}/4 \rfloor$.

Figure (6.3) shows the original image for this example. In figure (6.2) we compare the two reconstructed images obtained by sample block ALS and ALS. The compression ratio 4.8 is calculated by the following formula

$$ratio = \frac{IJK}{R_{est}(I+J+K)}.$$

The execution time for updating the factor matrices in the ALS is 5.86 seconds versus 1.52 seconds for the sample block ALS.

6.4. Conclusion

We propose a sampling block algorithm in order to determine the canonical polyadic decomposition of a third order tensor. The sampling method ensures the selection of the each rank-one component during the specific amount of iterations. The numerical experiments show cases of swamp reduction of ALS method. Our experiments also show the significant decrease in the execution time against the ALS algorithm. The performance of the algorithm was also tested on data compression. The future work is to find an effective way of sampling and to discuss the convergence of the algorithm.

LIST OF REFERENCES

[1] E. Acar, D. Dunlavy, T. Kolda, A scalable optimization approach for fitting canonical tensor decompositions, Journal of chemometrics, 25, pp. 67-86 (2011).

[2] E. Acar, C. A. Bingol, H. Bingol, R. Bro, and B. Yener, Multiway analysis of epilepsy tensors, Bioinformatics, 23 (13), pp. i10-i18, (2007).

[3] H. Attouch, J. Bolte, B. F. Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. Math. program. Ser. A 137. pp 91-129 (2013).

[4] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Lojasiewicz inequality, Math. Oper. Res., vol. 35 pp. 438457 (2010).

[5] A. Auslender, Optimisation. Methodes numeriques, Masson, Paris, New York, Barcelona (1976).

[6] H. H. Bauschke, P. L. Combettes, Convex Analysis and Monotone Operator Theory in Hilbert Spaces, Springer, Second edition, (2017).

[7] A. Beck, M. Teboulle, Gradient-based algorithms with applications to signal recovery problems, Convex optimization in signal processing and communications, Cambridge university press, pp. 42-88 (2010).

[8] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM journal on imaging sciences, Vol. 2, no. 1, pp. 183-202 (2009).

[9] D. P. Bertsekas. Nonlinear Programming. Second edition, Athena scientific press, (2008).

[10] J. Bolte, S. Sabach, M. Teboulle, Proximal alternating linearized minimization for nonconvex and nonsmooth problems, Math. Program, vol. 146, pp.459-494 (2014).

[11] T. Bouwmans, A. Sobral, S. Javed and S. K. Jung, and E.-H. Zahzah, Decomposition into Low-rank plus Additive Matrices for Background/Foreground Separation: A Review for a Comparative Evaluation with a Large-Scale Dataset, Computer Science Review, 23, pp. 1-71 (2017).

[12] S. Boyd, N. Parikh, E. Chun, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and trends in machine learning, vol. 3, no. 1, pp.1-122 (2011).

[13] J. Brachat, P. Comon, B. Mourrain and E. Tsigaridas Symmetric Tensor Decomposition, Linear Algebra and Applications 433, 11-12, pp. 1851-1872 (2010).

[14] E. K. P. Chong, S. H. Zak, An introduction to optimization, Fourth edition, John Wiley and Sons (2013).

[15] J. Carroll, J. Chang, Analysis of individual differences in multi-dimensional scaling via an n-way generalization of eckart-young decomposition, Psychometrika, Vol. 35(3), pp.283319 (1970).

[16] P. Comon, Tensor decompositions in Mathematics in Signal Processing V, J. G. McWhirter and I. K. Proudler, eds., Clarendon Press, Oxford, UK, pp. 1-24 (2002).

[17] P. Comon, G. Golub, L-H. Lim and B. Mourrain. Symmetric tensors and symmetric tensor rank. SIAM Journal on Matrix Analysis and Applications, 30 (3), 1254-1279, (2008).

[18] P. Comon and C. Jutten. Handbook of Blind Source Separation: Independent component analysis and applications. Academic press, (2010).

[19] L. De Lathauwer, J. Castaing and J-F. Cardoso Fourth-order cumulant-based blind identification of underdetermined mixtures. IEEE Transactions of Signal Processing, 55 (6), (2007).

[20] L. De Lathauwer, B. De Moor, J. Wandewalle, A multilinear singular value decompositions, SIAM J. Matrix Anal. Appl., 21, pp. 1253-1278 (2000).

[21] V. De Silva and L.-H. Lim Tensor rank and the ill-posedness of the best low-rank approximation problem, SIAM J. Matrix Anal. Appl., 30 , pp. 1084-1127,(2008).

[22] M. De Vos, A. Vergult, L. De Lathauwer, W. De Clercq, S. Van Huffel, P. Dupont, A. Palmini, and W. Van Paesschen, Canonical decomposition of ictal EEG reliably detects the seizure onset zone, Neuroimage, 37 (3), 844-854, (2007).

[23] I. Domanov and L.D. Lathauwer, Canonical polyadic decomposition of third-order tensors: reduction to generalized eigenvalue decomposition, SIAM J. Matrix Anal. Appl., 35, pp. 636-660 (2014).

[24] P. E. Frandesn, K. Jonasson, H.B. Nielsen, Unconstrained optimization, third edition, IMM (2004).

[25] G. Golub, C. Van Loan. Matrix computations. JHU press, Fourth edition, (2013).

[26] W. Greub, Multilinear algebra, second edition, Springer, (1978).

[27] J. Hiriart-Urruty, C. Lemarchal, Fundamentals of Convex Analysis, Springer, (2001).

[28] J. Hastad. Tensor rank is np-complete. Journal of Algorithms, 11(4), pp. 644 -654 (1990).

[29] A. Harshman, Foundations of the PARAFAC procedure: Models and conditions for an explanatory multi-model factor analysis, UCLA working papers in phonetics, Vol. 16, pp. 1-84 (1970).

[30] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, J. Math. Phys. Vol. 6, pp 164-189 (1927).

[31] F. L. Hitchcock, Multiple invariants and generalized rank of a p-way matrix or tensor, J. Math. Phys. Vol. 7 pp 39-79 (1927).

[32] C.J. Hillar and L. H. Lim, Most tensor problems are NP-hard, Journal of the ACM, 60, no. 6 (2013).

[33] K. Jackowski and B. Cyganek, A learning-based colour image segmentation with extended and compact structural tensor feature representation, Pattern Analysis and Applications, 20 (2) pp. 401414 (2017).

[34] S. Kaczmarz, Approximation solution of systems of linear equations, International journal of control, Vol. 57, no. 6, pp. 1269-1271 (1993).

[35] L. A. Kiers, Toward a standardized notation and terminology in multiway analysis, J. Chemometrics, Vol. 14, pp 105-122 (2000).

[36] T. G. Kolda, B. W. Bader. Tensor decompositions and applications. SIAM review. Vol. 51, No. 3. pp.455-500 (2009).

[37] T. G. Kolda, Multilinear operators for higher-order decompositions, Tech report SAND2006-2081, Sandia national laboratories, Albuquerque, NM, Livermore, CA (2006).

[38] K. Kurdyka, On gradients of functions definable on o-minimal structures. Ann. Inst. Fourier 48, pp. 769-783 (1998).

[39] P.M. Kroonenberg, Applied Multiway Data Analysis. Wiley (2008).

[40] W. Krijnen, T. Dijkstra, A. Stegeman, On the non-existance of optimal solutions and the occurence of degeneracy in the candecomp/parafac model, Psychometrika, 73 (3), pp. 431-439 (2008).

[41] J. B. Kruskal, statement of some current results about three-way arrays, manuscript, AT& T Bell Laboratories, Murray Hill (1983).

[42] J. B. Kruskal, Rank, decomposition, and uniqueness for 3-way and N-way arrays, in Multiway data analysis, R. coppi and S. Bolasco, eds. Amesterdam, pp 7-18 (1989).

[43] J. B. Kruskal, Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics, Linear algebra and its applications, Vol. 18(2), pp.95138 (1977).

[44] S. Lang, Algebra, Graduate text in mathematics, Springer, (2002).

[45] J.M. Landsberg, Tensors: Geometry and Applications, AMS, Providence, Rhode Island, (2010).

[46] K. Levenberg, A Method for the Solution of Certain Non-Linear Problems in Least Squares, Quarterly of Applied Mathematics Vol. 2 (2), 164168 (1944).

[47] K. Madsen, H. B. Nielson, O. Tingleff, Methods for non-linear least squares problems, second edition, Technical university of Denmark (2004).

[48] D. Marquardt, An Algorithm for Least-Squares Estimation of Nonlinear Parameters, SIAM Journal on Applied Mathematics, Vol. 11 (2): 431441 (1963).

[49] C. Navasca, L.D. Lathauwer and S. Kindermann, Swamp reducing technique for tensor decomposition, in the 16th Proceedings of the European Signal Processing Conference, (2008).

[50] S. Lojasiewicz, Sur la geometrie semi- et sous- analytique. Ann. Inst. Forier vol. 43, pp. 1575-1595 (1993).

[51] D. Nion, L. De Lathauwer, An enhanced line search scheme for complex-valued tensor decompositions, application in ds-cdma. Signal Processing, 88(3), pp. 749-755, (2008).

[52] J. Nocedal, S. J. Wright, Numerical optimization, Springer series in operation research, Second edition, (2006).

[53] N. Parikh, S. Boyd. Proximal algorithms. Foundations and trends. Vol. 1, No. 3. pp 123-231 (2013).

[54] P. Paatero, Construction and analysis of degenerate parafac models, Journal of chemometrics, Vol. 14, no. 3, pp. 285-299 (2000).

[55] P. Paatero, A weighted non-negative least squares algorithm for three-way PARAFAC factor analysis. Chemometrics and Intelligent Laboratory Systems, 38(2), pp. 223242 (1997).

[56] P. C. Parks, S. Kaczmarz (1895-1939), International journal of control, Vol. 57, no. 6, pp. 1263-1267 (1993).

[57] R. Penrose, A generalized inverse for matrices. Proceedings of the Cambridge Philosophical Society. Vol. 51 (3) (1955).

[58] M. J. D. Powell, On search directions for minimization algorithms, Math. Programming, vol. 4 pp. 193201 (1973).

[59] Y. Qi, P. Comon, Uniqueness of nonnegative tensor approximations, IEEE Transactions on Information Theory , 26 (4), pp. 2170-2183 (2016).

[60] R. T. Rockafellar, R. Wets, Variational analysis. Springer, Berlin (1998).

[61] N.D. Sidiropoulos, G.B. Giannakis, and R. Bro, Blind PARAFAC receivers for DS-CDMA systems, IEEE Trans. on Signal Processing, 48 (3), 810-823, (2000).

[62] N. Sidiropoulos, R. Bro, and G. Giannakis, Parallel factor analysis in sensor array processing, IEEE Trans. Signal Processing, 48, 2377-2388, (2000).

[63] N. D. Sidiropoulos, R. Bro, On the uniqueness of multi-linear decomposition of N-way arrays, Journal of chemometrics, Vol. 14, no. 3, pp. 229-239, (2000).

[64] A. Smilde, R. Bro, P. Geladi, Multi-way analysis: Applications in chemical sciences, Wiley, West Sussex, England (2004).

[65] A. Sobral, T. Bouwmans and E.-H. Zahzah, Robust Low-Rank and Sparse Matrix Decomposition: Applications in Image and Video Processing, CRC Press, Taylor and Francis Group, (2015).

[66] J. T. Sun, H. J. Zeng, H. Liu, Y. Lu, Z. Chen, Cubes SVD: A novel approach to personalized web search, in WWW 2005: Proceeding of the 14th international conference on world wide web, ACM press, pp 382-390 (2005).

[67] P. Tseng and S. Yun, A coordinate gradient descent method for nonsmooth separable minimization, Math. Program., vol. 117 , pp. 387423 (2009).

[68] J. M. F. Ten Berge, The typical rank of tall three-way arrays, Psychometrika, Vol. 65, pp. 525-532 (2000).

[69] J. M. F. Ten Berge, Partial uniqueness in CANDECOMP/ PARAFAC, J. Chemometrics. Vol. 18, pp. 12-16 (2004).

[70] J. M. F. Ten Berge, H. A. L. Kiers. Simplicity of core arrays in three-way principal component analysis and the typical rank of $p \times q$ arrays, Linear algebra Appl., Vol. 294, pp. 169-179 (1999).

[71] J. M. F. Ten Berge, Kruskal's polynomial for $2 \times 2 \times 2$ arrays and a generalization to $2 \times n \times n$ arrays, Psychometrika, Vol. 56, pp. 631-636 (1991).

[72] G. Tomasi, Use of the properties of the Khatri-Rao product for the computation of Jacobian, Hessian, and gradient of the PARAFAC model under MATLAB,

private communication, 2005.

[73] L. R. Tucker, Implications of factor analysis of three-way matrices for measurement of change, Problems in measuring change, pp. 122137, (1963).

[74] L. R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika, 313, pp.279-311 (1966).

[75] C. F. Van Loan, The ubiquitous Kronecker product, J. Comput. Appl. Math. 123. pp. 85-100 (2000).

[76] L. Vandenberghe, Fast proximal gradient methods, (2010).

[77] X. Wang, C. Navasca. Lowrank approximation of tensors via sparse optimization, Numerical Linear Algebra with Applications, vol. 25 (2018).

[78] X. Wang, C. Navasca, S. Kindermann, On Accelerating the Regularized Alternating Least Square Algorithm for Tensors, Electronic Transactions on Numerical Analysis, Vol. 48, pp. 114 (2018).

[79] Y. Xu, W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM J. imaging science. Vol. 6, No. 3. pp. 1758-1789 (2013).

[80] Y. Xu, L. Yu, H. Xu, H. Zhang, and T. Nguyen, Vector Sparse Representation of Color Image Using Quaternion Matrix Analysis, IEEE TRANSACTIONS ON IMAGE PROCESSING, 24 (4) , pp. 1315-1328 (2015).